SMAF 仕様書

Synthetic music Mobile Application Format

Ver. 3.05

ヤマハ株式会社

Ver.	Date	内容
3.00	2001. 05. 24	SMAF v3.00 仕様 SMAF v2.04 から改版してリリースv2.04 からの主な変更点 演奏用 Track に新しい Format Type を定義して従来の運用と区別する仕様とした 表示用 Track に一文字色替えの機能を追加した
3.01	2001.06.05	Contents Info Chunk および Optional Data Chunk にタグ "MI"を追加 Optional Data Chunk 仕様変更 サブ Chunk を追加 Graphics Track の Banner Info の Effective Span の特別な値の解釈の説明の誤りを修正
3.02	2001.06.28	Huffman 符号化の説明の Event へ移動 Sequence Data Chunk のハフマン符号化適用について変更 Exclusive Messege の Data Size 説明の修正 End of Sequence において Handy Phone Standard と Mobile Standard との違いを明記 Note Messege 項目に Stream PCM Data Chunk の Wave コントロールについて追記 4.3 タグに ES と VC を追加 6.3.2 座標指定の解釈について文章を追加 6.3.6 1) Bmp Tile の Event Type 誤記訂正 8.3.1.-6 リハーサルマークの説明を一部削除
3.03	2001.09.25	Stop Point における gatetime の再生保証を変更 Note Number に関する説明を追加 標準音色マップを Handy phone standard と Mobile standard で分ける
3.04	2001.12.03	Expressionの標準タイプと短縮タイプの対応値変更
3.05	2002.03.13	本文書名を「SMAF」から「SMAF 仕様書」に変更 イベントの同時実行について表現を「値が"0"のデュレーションを挟んで連続した」に変更 SMAF ファイル概念図に Optioal Data Chunk を追加 コンテンツタイプの範囲を追加 Body 部の説明図を追加 Note Message と Start Point と Stop Point の関係に実装依存の部分を明記 Expression の短縮タイプに対する標準タイプの値を変更 Sequence Data Chunk の Event における States Byte の表を追加 Chunk ID & TAG 一覧に Mtsp,Mwa*を追加 4.2.3 韓国語 ISO-2022-KR から EUC-KR(KS) へ修正 4.2.4 CopyStatus の不使用ビットを0埋めから1埋めに変更。 6.1 3) 韓国語 ISO-2022-KR から EUC-KR(KS) へ修正

1. はじめに

SMAF は携帯端末用マルチメディアコンテンツを小さなサイズで効率よく表現することを目的に設計したデータフォーマット仕様である。
SMAF 仕様に基づいたデータは、1 つ以上のシーケンスデータを束ねた形式となっている。
各シーケンスデータは、再生デバイスに対応したデータとなっている。
全てのシーケンスは、再生時に時間軸を共有する。つまり、全シーケンスは、時間的に同期して再生される。
SMAF は、こうした同期再生コンテンツを表現するためのデータ形式である。

SMAF の基本部分は弊社携帯端末用音源 LSI 向けの演奏データ表現として設計したものである。また SMAF は、柔軟性と拡張性を備えており、将来的な機能拡張にも対応できる設計となっている。
本書は SMAF のデータ表現形式についての仕様書である。本書ではデータ表現とその基本的な意味について定義している。SMAF データを再生するための実装仕様や、SMAF データの作り方を定義するのが目的としていない。
SMAF の再生系の実装仕様および SMAF データ制作ガイドラインやデータの運用ルールについては、それぞれ別のドキュメントとして規定する必要がある。

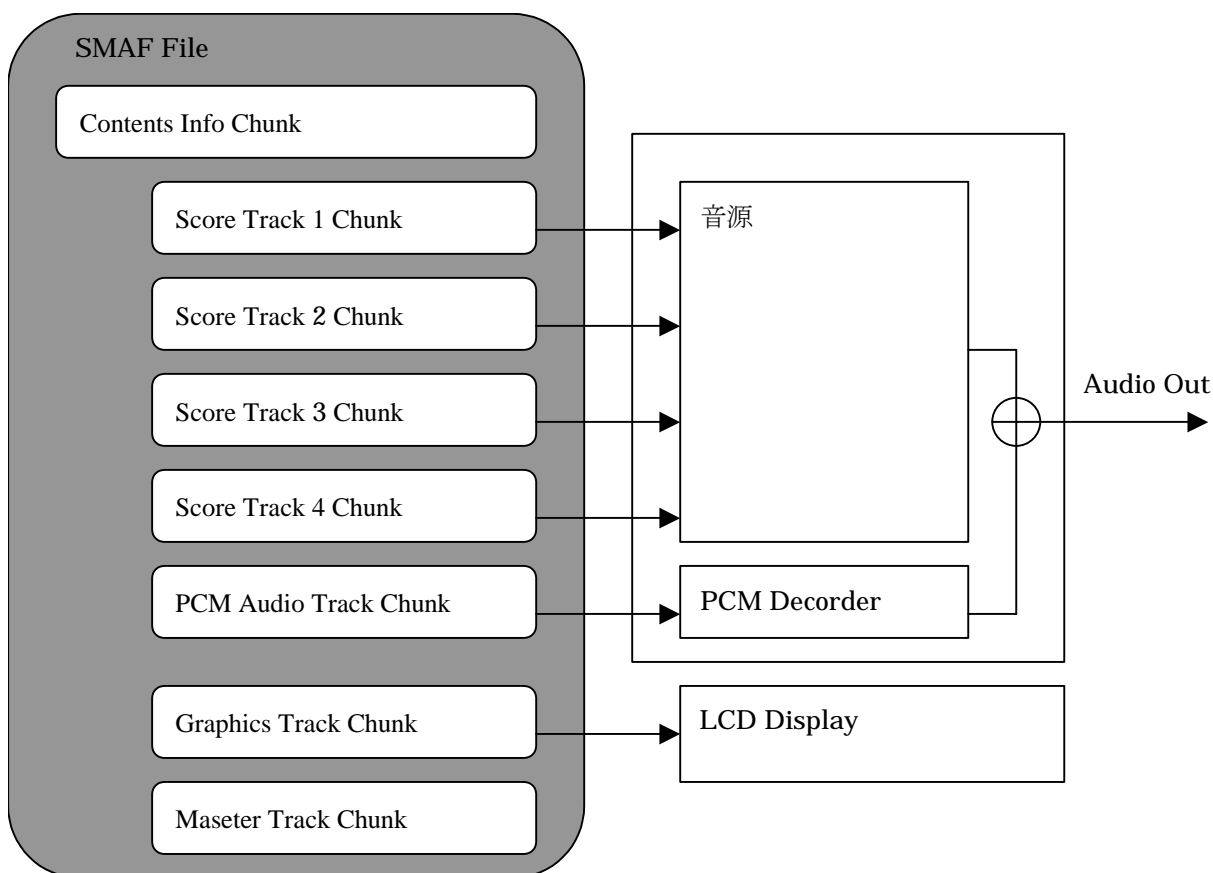
2. 基本概念

2.1. SMAF 仕様が定義するもの

SMAF 仕様はファイルのデータ表現形式を定義したものである。SMAF 仕様に合致するように作られたファイルを SMAF ファイルと呼ぶことにする。
SMAF ファイルの中身は、大きく分けて SMAF ファイルの管理用の情報とシーケンスデータの集合に分類できる。

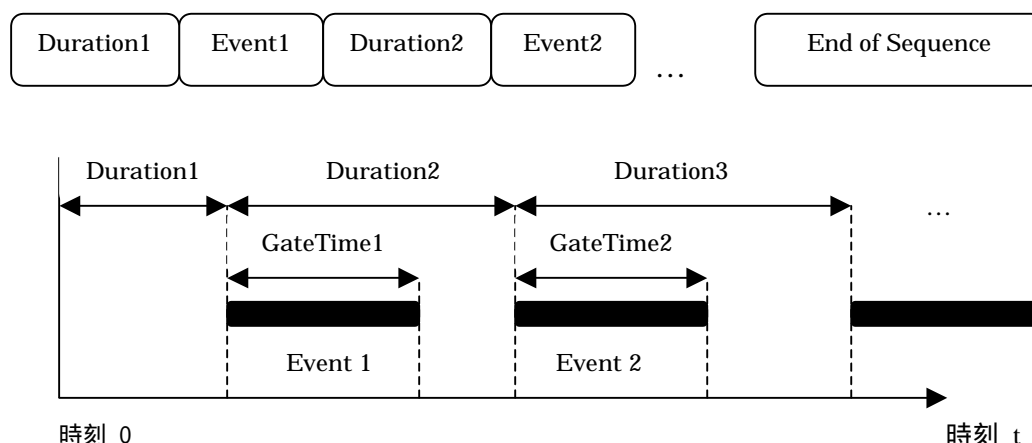
SMAF ファイルの管理用情報は Contents Info Chunk に格納する。詳細はデータフォーマット定義のページで詳しく説明する。
出力デバイス毎に定義したシーケンスデータの集合は、最低1つのシーケンスデータを含む。ここで言う出力デバイスは、シーケンスデータ毎に定義された論理的なデバイスで、通常は対応するハードウェアが仮定されているが、必ずしも1対1に対応するとは限らない。

シーケンスデータとは、出力デバイスに対する制御を時間を追って定義したデータ表現である。1つの SMAF ファイルに含まれる全てのシーケンスデータは時刻0で同時に再生を開始するものと定義し、結果的に全てのシーケンスデータが同期して再生されることが表現されている。

SMAF 仕様で定義している出力デバイスとして、MIDI 相当の制御データで発音を行う音源デバイス、PCM データの再生を行う PCM 音源デバイス、テキストや画像の表示を行う表示デバイスがある。

Track と出力デバイスの対応概念図

2.2. シーケンスデータ

シーケンスデータはイベントとデュレーションの組み合わせで表現する。
イベントとはシーケンスデータに定義してある出力デバイスに対する制御内容のデータ表現である。
デュレーションはイベントとイベントとの間の経過時間を表現する。イベントの処理時間は実際には”0”ではないが、SMAF のデータ表現としては”0”とみなし、時間の流れは全てデュレーションで表すことにする。あるイベントを実行する時刻は、そのシーケンスデータの先頭からのデュレーションを積算することで一意に決定できる。イベントの処理時間は次のイベントの処理開始時刻に影響しない、という原則である。従って、値が”0”のデュレーションを挟んで連続したイベントは同時に実行すると解釈する。

イベントの処理時間は常に”0”とみなすが、イベント自体が内部にサブシーケンスを含んでいる場合がある。例えば、Note イベントの GateTime の処理がそれに当たる。Note イベントが表現する内容は、イベントの実行時刻で NoteON 処理を瞬時に行い、GateTime 経過後 NoteOFF 処理を行う、というものである。この場合の GateTime は Note イベントの内部的な時間経過を表しており、次のイベントの実行時刻には影響していない。より複雑なサブシーケンスを内部に抱えているイベントもあるが、この原則は同じである。

Sequence Data 概念図

2.3. 時間軸の定義

シーケンスデータを解釈するにあたり、時間の流れ(時間軸)の定義として以下の3つが考えられる。

A) 日常生活で使っている時間軸。伸び縮みしない。単位は秒。絶対時間。
B) シーケンスデータを再生するシーケンサが管理する時間軸。再生速度を変更することにより絶対時間に対して伸び縮みする。単位は秒。再生系時間。
C) 主に音楽演奏を表現するときに使う時間軸。楽譜表現に対応し、演奏テンポを変更することで再生系時間に対して伸び縮みする。単位は拍。

SMAF における時間経過は全て B の再生系時間で表現する。

SMAF はコンテンツ再生に主眼を置いた設計をしているため再生速度変更の概念はあるが、テンポ変更の概念がない。そのため C の時間表現は敢えて採用しない。
また、シーケンスデータ毎に時間表現の単位となる長さを定義することでデータ表現の精度とデータ量のバランスがとれるように考慮している。

2.4. トラック

トラックは SMAF ファイルの基本構成要素となるデータ表現で、シーケンスデータとその動作環境を定義している。

トラックと出力デバイスの関係は以下の通りである。

SCORE Track := 音源デバイス
PCM Track := PCM 音源デバイス
GRAPHICS Track := 表示デバイス
MASTER Track := SMAF シーケンサ自身

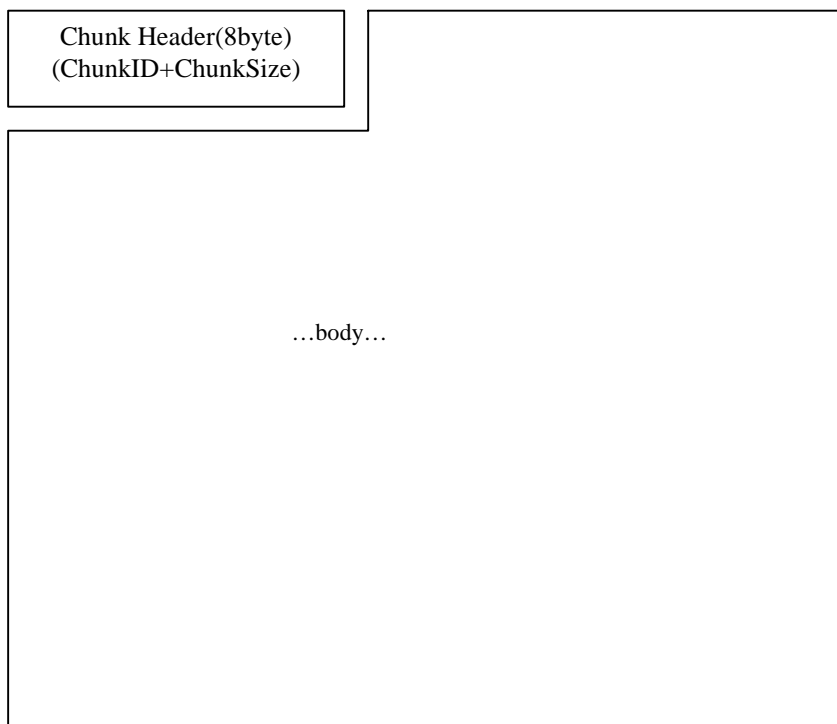
2.5. SMAF 再生系実装ガイドライン

SMAF のイベントの実行時刻や処理内容は厳密に定義されており、曖昧さが入り込む余地はほとんどない。とはいえ、実際の実装では SMAF で表現した通りの厳密な再生は不可能で、必ず実行時間の誤差や、イベント解釈の振れが発生する。また、イベントによっては実装自体が不可能な場合もある。これらの実行時間誤差や解釈の振れ、実装の限界についての許容限度は別途、再生系実装ガイドラインとして定義されるべき性質のもので、本仕様書では触れないこととする。

3. データ表現概要

3.1. Chunk 構造

SMAF データの基本構造は Chunk と呼ぶデータの固まりである。

以下に Chunk の構造を示す。

Chunk 構造図

Chunk は 8 バイトの Header 部と任意長の Body 部とに分けられる。Header 部は更に4バイトの Chunk ID と4バイトの Chunk Size に分けられる。
Chunk ID は Chunk の識別子に用い、Chunk Size は Body 部の長さ(バイト単位)を表現している。

3.1.1. Chunk ID

Chunk ID はファイル内でユニークな値を持つ。通常4byte の ASCII 文字を ID としてもち、大文字と小文字を区別する。例えば、Chunk ID = "ABCD" の場合、下表となる。

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	0x41 (“A”)							
Data#1	0x42 (“B”)							
Data#2	0x43 (“C”)							
Data#3	0x44 (“D”)							

Track Chunk ID においては、先頭 3byte でどの Track であるかを表わし、最後の1byte で Track Number を Binary 値で表現する。

3.2. ChunkSize

Chunk Size は Body の byte カウントであり、4byte で記す。
ただし、File Chunk の Body の byte カウントは CRC を含める。例えば、size = 0x12345678 の場合は、下表となる。

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	0x12							
Data#1	0x34							
Data#2	0x56							
Data#3	0x78							

3.3. File Chunk

File Chunk は SMAF ファイル全体を表す Chunk である。ファイル Chunk の Body 部は Chunk 列で、最後に2バイトの CRC が付加されている。
Chunk 列の先頭は必ず Contents Info Chunk で、その後は1個以上のトラック Chunk が続く。Chunk 列の中に同じ ChunkID を持つ Chunk が存在してはいけない。

最後の CRC はファイル Chunk が壊れていないかどうかを確認するために用いる。

3.4. Track Chunk

Track Chunk には Score,PCM Audio,Graphics,Master のグループがある。
Track Chunk の Chunk ID の下位1バイトはトラック番号となっているため、それぞれのトラックは最大 256トラックまで表現することができる。

以下は現在定義されている Track Chunk の Chunk ID である。

Chunk ID	"MTR*"	:Score	Track	* = 0x00 ~ 0xFF
Chunk ID	"ATR*"	:PCM Audio	Track	* = 0x00 ~ 0xFF
Chunk ID	"GTR*"	:Graphics	Track	* = 0x00 ~ 0xFF
Chunk ID	"MSTR"	:Master	Track	

3.4.1. Track Chunk の種類

- Score Track Chunk
  音源を再生するためのシーケンス・データを格納する。

- PCM Audio Track Chunk
  ADPCM や MP3, TwinVQ 等の PCM 系 Audio 発音をイベント形式で格納する。

- Graphics Track Chunk
  背景画や差込静止画とテキストデータと、これらを再生するためのシーケンス・データを格納する。

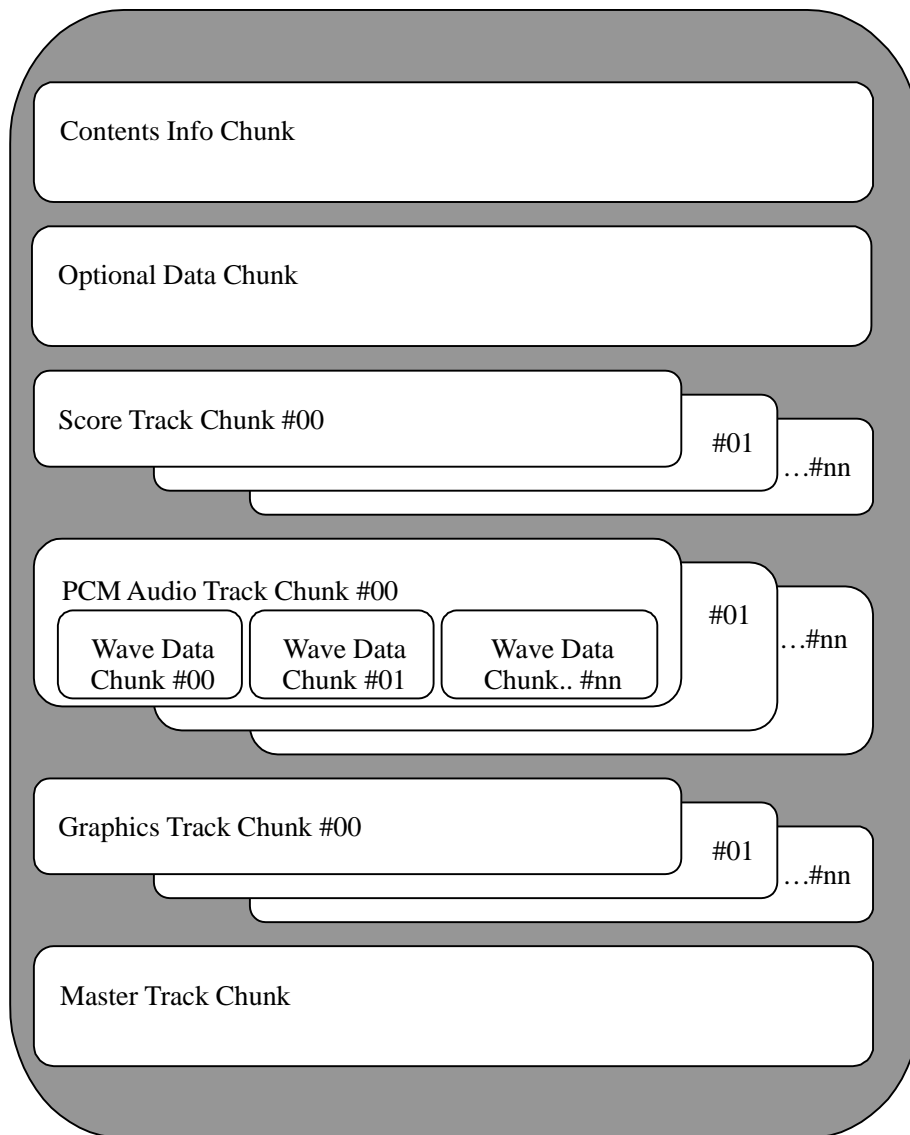
- Master Track Chunk
  Score Track などの再生シーケンスに同期した音楽情報シーケンスや、SMAF 再生系を制御するためのシーケンス・データを格納する。

Master Track Chunk を除く Track Chunk は Chunk ID 内部に Track Number を持ち、各 256トラック分記述可能である。

Body 部の大きさは任意に決めることが可能である。Body 部に更に別の Chunk を定義することで Chunk を入れ子にすることも可能である。Body 部のフォーマットは Chunk ID 毎に定義する。

SMAF におけるデータ表現の最小単位はバイトである。本仕様書の説明では原則としてバイト単位でデータ構造の定義を行っている。複数バイトに跨って整数値を格納する場合は、特に断りのない限り BigEndian 表現(上位バイトを先に配置)である。

3.5. SMAFファイルの概念図

以下は SMAF ファイルの概念図である。

SMAF ファイル概念図

SMAF 仕様としては、各々のトラック数の上限やトラック列の並べる順番についての規定を行わない。
これらは SMAF の運用ルールとして別途規定する。

4. データ・フォーマット(基本機能)

SMAF の基本のデータフォーマットについて、具体的に記述する。定義項目中で「Reserved」になっているものは、今後別途定義する予定があるので使用は禁止となる。

4.1. File Chunk

File Chunk ID として

Chunk ID	"MMMD"	:Mobile Application Data Chunk

を定義する。File Chunk のみ Body の最後に CRC を付加する。

4.1.1. CRC

Chunk Header 及び Body の Byte 列に対し、下記に示す割数で割り算した余りを CRC 値とする。ただし、割り算途中の引き算の代わりにビット毎の排他的論理和を用いる。

割数 : 0x11021 (CCITT X.25 による)

(*) APPENDIX にサンプル・コードを記載する。

4.2. Contents Info Chunk

Chunk ID	"CNTI"	:Contents Information Chunk

body に以下の順でデータを格納する。

- Contents Class :1 byte(必須)
- Contents Type :1 byte(必須)
- Contents Code Type :1 byte(必須)
- Copy Status :1 byte(必須)
- Copy Counts :1 byte(必須)
- Option :n byte(Option)

4.2.1. Contents Class

コンテンツのクラスを表現する。

Contents Class	Description
0x00	YAMAHA
0x01〜0xFF	Vender ID

※PDA 端末等、将来の多機能端末で同様のデータフォーマットまで流用する場合の区別に用いる。

4.2.2. Contents Type（Contents Class 0x00 用）

コンテンツのタイプを表現する。

Contents Type	Description
0x00～0x0F, 0x30～0x33	着信メロディ
0x10～0x1F, 0x40～0x42	カラオケ系
0x20～0x2F, 0x50～0x53	CM系
上記以外の未使用値	Reserved

4.2.3. Contents Code Type

Contents Info Chunk の Option 内のデータ表現で使用する文字コード体系を特定する。

Contents Code Type	Description	Language
0x00	Shift-JIS	日本語
0x01	ISO 8859-1 (Latin-1)	英語、フランス語、ドイツ語 イタリア語、スペイン語、ポルトガル語
0x02	EUC-KR(KS)	韓国語
0x03	HZ-GB-2312	中国語(簡体字)
0x04	Big5	中国語(繁体字)
0x05	KOI8-R	ロシア語など
0x06	TCVN-5773:1993	ベトナム語
0x07～0x1F	Reserved	Reserved
0x20	UCS-2	Unicode
0x21	UCS-4	Unicode
0x22	UTF-7	Unicode
0x23	UTF-8	Unicode
0x24	UTF-16	Unicode
0x25	UTF-32	Unicode
0x26～0xFF	Reserved	Reserved

ただし、Contents Info Chunk の Option においては、どのタイプも『,(0x2C)』は Option のデリミタとして定義しているため、エスケープキャラクタとしてのバックスラッシュ『¥(0x5C)』と合わせて使用する。

表記	機能
¥,	“,” を表す
¥¥	¥を表す
¥	無視する

デリミタを正しく識別出来ないことが想定される文字コードが設定されている場合には Contents Info Chunk の Option にデータを記述せず、Optional Data Chunk にデータを記述するものとする。

4.2.4. Copy Status（Contents Class 0x00, Contents Type 0x00 ~ 0xFF 用）

コンテンツの持つコピー定義を表現する。

	b7	b6	b5	b4	b3	b2	b1	b0
不可 bit	Reserved	Reserved	Reserved	Reserved	Reserved	編集	保存	転送

b0 は転送の可/不可、b1 は保存の可/不可 b2 は編集の可/不可を定義する。(0:可　1:不可)

Reserved bit は"1"で埋める。

4.2.5. Copy Counts（Contents Class 0x00, Contents Type 0x00 ~ 0xFF 用）

コピー回数を表現する。

Copy Counts	Description
0x00～0xFE	Copy Counts
0xFF	Copy Counts(255 以上)

※ コピー／移動が発生するたびに1カウント進める。

4.2.6. Option（Contents Type 0x00, Contents Class 0x00 ~ 0xFF 用）

ジャンル名、曲名、アーティスト名、作詞／作曲者名等を格納する。必ずしも表示に使用するものではなく、個別データの認識に使用するものである。

データは可変長とし、『タグ (2byte)』+『 : (0x3A)』+『Data』+『 , (0x2C)』で区切って記述する。タグ名を以下とする。タグは 2byte 固定のバイト列とする。

『Data』内において『 , (0x2C)』を文字として使用する場合 のエスケープキャラクタを§4.2.3に定義する。

名称	タグ名	Hex
ベンダー名	VN	0x56 0x4E
キャリア名	CN	0x43 0x4E
カテゴリー名	CA	0x43 0x41
曲名	ST	0x53 0x54
アーティスト名	AN	0x41 0x4E
作詞	WW	0x57 0x57
作曲	SW	0x53 0x57
編曲	AW	0x41 0x57
Copyright©	CR	0x43 0x52
管理者団体名	GR	0x47 0x52
管理情報	MI	0x4D 0x49
作成日時	CD	0x43 0x44
更新日時	UD	0x55 0x44

Unicode の追加にあたり、Option 内のデリミタを識別できない文字コードが存在するため

Optional Data Chunk を追加した。

4.3. Optional Data Chunk

Chunk ID "OPDA" :Optional Data Chunk

ジャンル名、曲名、アーティスト名、作詞／作曲者名等を格納する。必ずしも表示に使用するものではなく、個別データの認識に使用するものである。
Optional Data Chunk の Body 部には Data Chunk 等の Sub Chunk 列を格納する。

1.) Data Chunk

Chunk ID "Dch*" :Data Chunk * = 0x00 ～ 0xFF

Data Chunk の Chunk ID の4byte 目は 格納する Data に応じて下に記述された Code Type の値を格納する。格納 Data はその Code Type で指定された文字コード体系で示されているものとする。

{
- タグ :2byte (固定)
- サイズ :2byte (固定)
- Data :nbyte (可変)

}
繰り返し

データは可変長とし、『タグ (2byte 固定)』+『サイズ(2byte 固定)』+『Data(可変長)』を一組として記述する。サイズは Data のサイズを指定する。

Optional Data Chunk の Data に使用する文字コード体系を特定する。

Code Type	Description	Language
0x00	Shift-JIS	日本語
0x01	ISO 8859-1 (Latin-1)	英語、フランス語、ドイツ語 イタリア語、スペイン語、ポルトガル語
0x02	ISO-2022-KR	韓国語
0x03	HZ-GB-2312	中国語(簡体字)
0x04	Big5	中国語(繁体字)
0x05	KOI8-R	ロシア語など
0x06	TCVN-5773:1993	ベトナム語
0x07～0x1F	Reserved	Reserved
0x20	UCS-2	Unicode
0x21	UCS-4	Unicode
0x22	UTF-7	Unicode
0x23	UTF-8	Unicode
0x24	UTF-16	Unicode
0x25	UTF-32	Unicode
0x26～0xFE	Reserved	Reserved
0xFF	Octet Stream	Binary 値

記述する文字コードが Unicode の場合、それぞれの文字群先頭に BOM（バイトオーダーマーク）を設定すること。BOM 無しの場合、ビッグエンディアンとして解釈する。

タグは以下のように定義する。

名称	タグ名	Hex
ベンダー名	VN	0x56 0x4E
キャリア名	CN	0x43 0x4E
カテゴリー名	CA	0x43 0x41
曲名	ST	0x53 0x54
アーティスト名	AN	0x41 0x4E
作詞	WW	0x57 0x57
作曲	SW	0x53 0x57
編曲	AW	0x41 0x57
Copyright©	CR	0x43 0x52
管理者団体名	GR	0x47 0x52
管理情報	MI	0x4D 0x49
作成日時	CD	0x43 0x44
更新日時	UD	0x55 0x44
編集後ステータス	ES	0x45 0x53
vCard	VC	0x56 0x43

4.4. Score Track Chunk

音源へ送り込む楽曲のシーケンス･トラックを格納する Chunk である。フォーマット認識用の1byte Data とトラック依存情報、及び複数のサブ Chunk で構成する。各サブ Chunk は、Score Track Chunk 内でそれぞれ1つしか持たない。

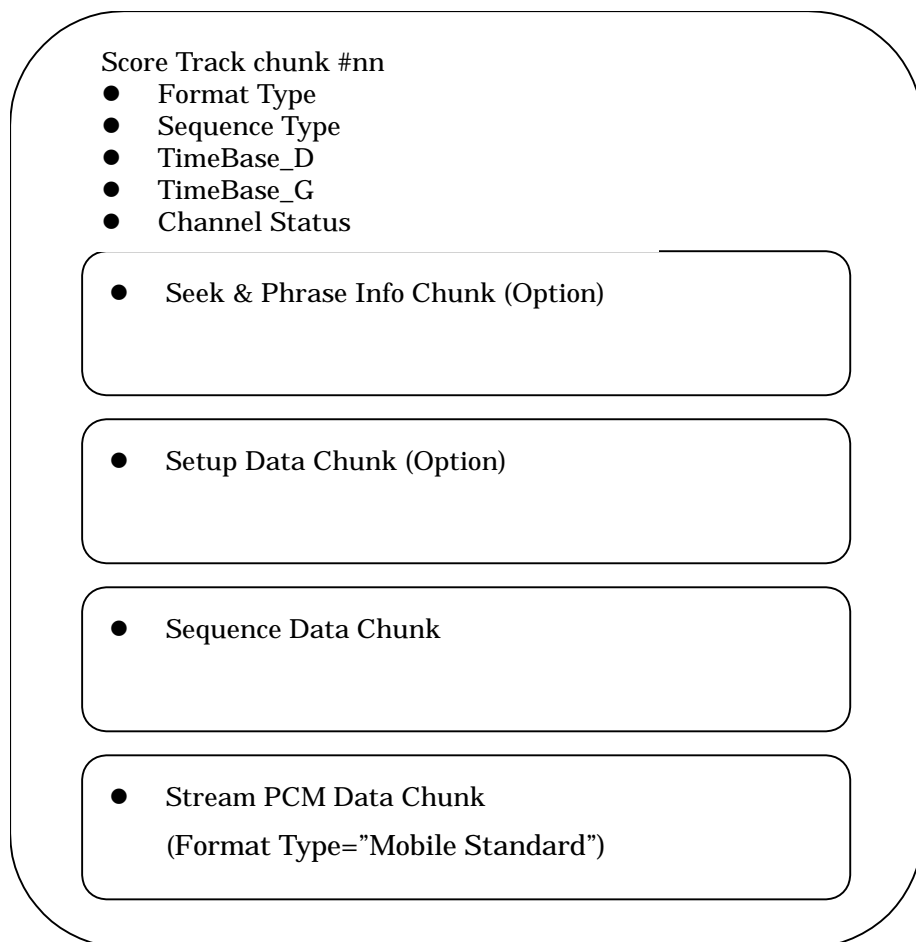
- Format Type : 1 byte（必須）
- Sequence Type : 1 byte（必須）
- TimeBase_D : 1 byte（必須）
- TimeBase_G : 1 byte（必須）
- Channel Status : n byte（必須）(Format Type に依存)
- Seek & Phrase Info Chunk : n byte（Option）
- Setup Data Chunk : n byte（Option）
- Sequence Data Chunk : n byte（必須）
- Stream PCM Data Chunk : n byte（Option）（Format Type=”Mobile Standard”の場合のみ）

Score Track chunk #nn
- Format Type
- Sequence Type
- TimeBase_D
- TimeBase_G
- Channel Status

- Seek & Phrase Info Chunk (Option)

- Setup Data Chunk (Option)

- Sequence Data Chunk

- Stream PCM Data Chunk (Format Type=”Mobile Standard”)

Format Type について

このステータスでこの Track Chunk の実フォーマットを定義する。データ・サイズ削減のため、LSI Native Format での記述や、将来パワフルな Control CPU を想定して、その他のシーケンス・フォーマットを記述可能とする。Compress はハフマン符号化による圧縮を行うことと定める。
(Compress については 4.4.2 Mobile Standard Format で説明する。)

Format Type	Description
0x00	Handy Phone Standard
0x01	Mobile Standard (Compress)
0x02	Mobile Standard (No Compress)
0x03～0xFF	Reserved

4.4.1. Handy Phone Standard（FormatType=0x00）

数世代に渡る Handy Phone で、スケーラブルに対応できるデータ・フォーマットである。データ制作に於いては、実装系の処理能力によって制限（制作ガイドラインの設定）を行う。同様に実装系についても実装ガイドラインが必要で、これら2つのガイドラインと Format が三位一体となって初めて実動作保証が行われる。

2.) Sequence Type

Sequence Data の記述には2種類を想定する。

- 0x00 :Stream Sequence
  Sequence Data は1つの連続したシーケンス・データである。Seek Point や Phrase List はシーケンス中の意味のある位置を外部から参照する目的で利用する。
- 0x01 :Sub-sequence
  Sequence Data は複数のフレーズデータを連続で表記したものである。Phrase List は外部から個別フレーズを認識する為に用いる。
- 0x02 ~ 0xFF（reserved）

3.) TimeBase_D, TimeBase_G

Sequence Data 内部で使用する基準時間を表現する。
TimeBase_D が Duration の基準時間、TimeBase_G が GateTime の基準時間。

Timebase_D,G	Description
0x00	1 msec
0x01	2 msec
0x02	4 msec
0x03	5 msec
0x04～0x0F	Reserved
0x10	10 msec
0x11	20 msec
0x12	40 msec
0x13	50 msec
0x14～0xFF	Reserved

4.) Channel Status

4Channel ある Sequence data の、それぞれの Channel の status 情報を格納する。 4 bit で 1 Channel 分の status 情報を格納し、各 Channnel について下表のように定義する。

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	Channel 0				Channel 1			
	KCS	VS	Ch Type		KCS	VS	Ch Type	
Data#1	Channel 2				Channel 3			
	KCS	VS	Ch Type		KCS	VS	Ch Type	

- Key Control Status(KCS)

Key Control のリクエストを受けた時、該当 Channel に対し、Key Control を行うか否かの指定をする。ON で Key Control を有効とする。

KCS	Description
0x0	OFF
0x1	ON

- Vibration Status(VS)

Vibration Control のリクエストを受けた時、該当 Channel に対し、Vibration を行うか否かの指定をする。ON で Vibration を有効とする。

VS	Description
0x0	OFF
0x1	ON

- Ch Type

該当 Channel に対し、Channel Type を指定する。

Ch Type	Description
0x0	No Care
0x1	Melody
0x2	No Melody
0x3	Rhythm

5.) Seek & Phrase Info Chunk

Sequence Data 内の意図的なポジションからスタート／ストップさせるため、又は区間リピート等を表現可能とする為の情報を格納する。この運用、及び機能はオプションであり、機種やデータを作成するときの意図に依存する。

Chunk ID	"MspI"	:Seek & Phrase Info

body に以下の順でデータを格納する。

- Start Point :1 or 4 byte（Option）
- Stop Point :1 or 4 byte（Option）
- Phrase List :n byte（Option）
- Sub-sequence List :n byte（Option）

Chunk が存在する以上、どれか一つのデータが存在すること。上記バイト・サイズにはタグ、コロン、カンマは含まない。各データ可変長であるので 『タグ(2byte)』+『:(0x3A)』+『Data』+『,(0x2C)』で区切って記述する。

a) Start / Stop Point

Sequence Data 又は Sub-sequence List 内の任意の位置を Start, Stop Point として指定を行う。タグは以下の通りである。タグは ASCII 文字とする。

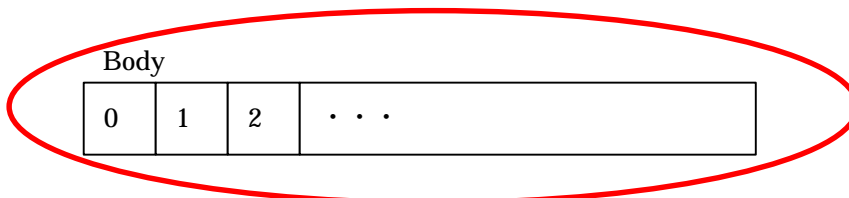
名称	タグ名	Hex
Start	st	0x73 0x74
Stop	sp	0x73 0x70

- Sequence Type = 0x00(Stream Sequence) の場合

Start Address : 4 byte
Stop Address : 4 byte

とし、タグの後に記述する。
各アドレス指定は、Sequence Data chunk の Body 部先頭を"0"とし、そこからのオフセットで記述する。

Body

0	1	2	・・・

また、各アドレス指定値は Duration の先頭を指し示す値である必要がある。

- Sequence Type = 0x01(Sub-sequence) の場合

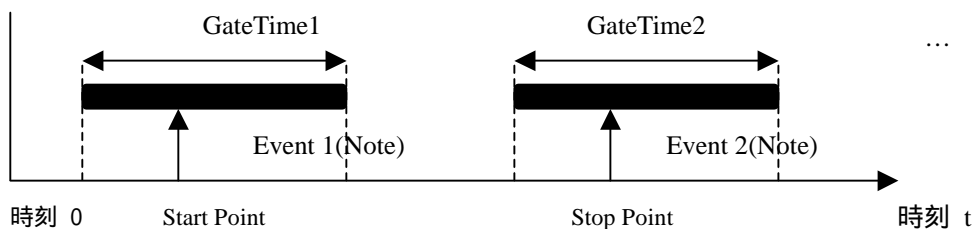
Start Address : 1 byte
Stop Address : 1 byte

とし、タグの後に記述する。
各アドレス指定は Sub-sequence List 先頭からの byte count 数である。
省略された場合は Start = 0, Stop = "File End" or "Sub-sequence List End" とする。

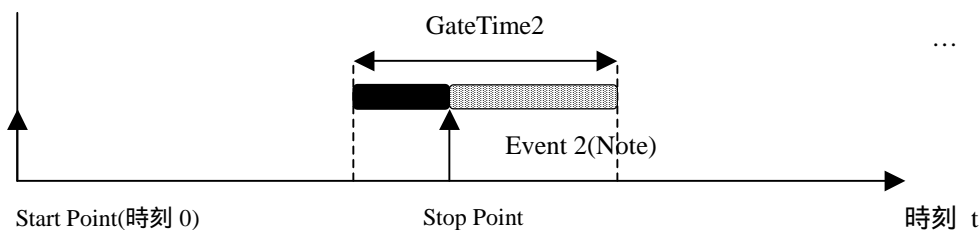
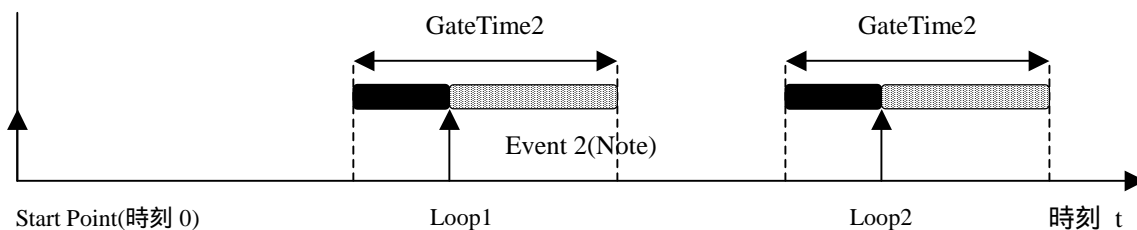
(*) Note Message と Start Point と Stop Point の関係

Note Message の発音開始時刻と Start Point 及び Stop Point の時刻を比較し、
Start Point の場合は
（Start Point の時刻）≦(Note Message の発音開始時刻)
であれば該当 Note　Message の発音を行う。
Stop Point の場合は
(Note Message の発音開始時刻) ＜（Stop Point の時刻）
であれば該当 Note Message の発音を行う。(gatetime の再生保証は実装依存とする)

また、Start　Point と Stop　Point の区間を loop する場合 loop 時の Stop　Point と次の Start　Point の時刻が同一となることを保証すること。

例として以下のように Sequence Data Chunk にて表現している場合を考える。

Start から Stop を再生する場合の Stream Sequence としての解釈は次の図のようになる。
<単一再生＞

<Loop 再生＞

…

loop1 = Stop Point – Start Point
loop2 =loop1 + (Stop Point – Start Point)

この部分の再生保証は実装依存とする

b) Phrase List

Sequence Data 内の任意の位置（区間）をフレーズ区間として指定する。Sequence Type の違いで使用用途が異なる。

Sequence Type = 0x00(Stream Sequence) の場合
Aメロ、Bメロ、サビ..等の Sequence Data の一部分（区間）への意味付けとして使用する。活用の仕方等はアプリケーションに依存する。

- Sequence Type = 0x01(Sub-sequence) の場合
  後述する『Sub-sequence List』内で用いる Phrase 区間として指定する。

データ表現はタグ名の後に

Start Address　　: 4 bytes
Stop Address　　: 4 bytes

を連続して記述する。各アドレス指定は Sequence Data Chunk の Body の先頭からの byte count である。
各区間の Start Address 及び Stop Address は任意に設定することができる。
各フレーズ区間を示すタグ名を以下に示す。タグは ASCII 文字とする。

名称	タグ名	Hex
a	Pa	0x50 0x61
b	Pb	0x50 0x62
c	Pc	0x50 0x63
d	Pd	0x50 0x64
e	Pe	0x50 0x65
:	:	:
z	Pz	0x50 0x7A
A(A メロ)	PA	0x50 0x41
B(B メロ)	PB	0x50 0x42
E(エンディング)	PE	0x50 0x45
I(イントロ)	PI	0x50 0x49
K(間奏)	PK	0x50 0x4B
S(サビ)	PS	0x50 0x53
R(リフレイン)	PR	0x50 0x52

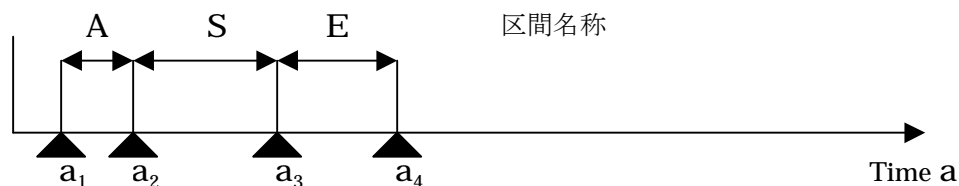
c) Sub-sequence List

Sequence Type が 0x01(Sub-sequence) の場合、Phrase List で指定されたフレーズ区間の再生順を記述する。Sequence Type が 0x00(Stream Sequence) の場合は無視する。
区間名称(a ～ z,A,B,E,I,K,S) を ASCII コードで Byte 単位で順番に記述する。タグは以下の通りである。タグは ASCII 文字とする。

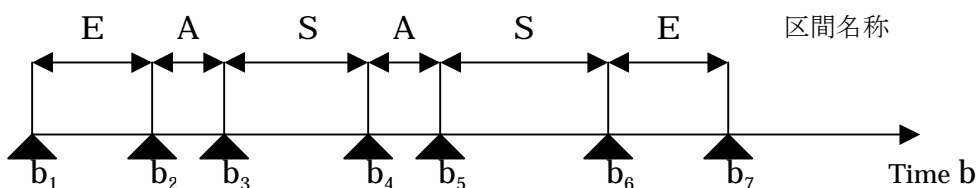
名称	タグ名	Hex
Sub-sequence List	SL	0x53 0x4C

(*)Subsequence List の解釈

Subsequence List を、Stream Sequence であらわした場合の例を下図に示す。

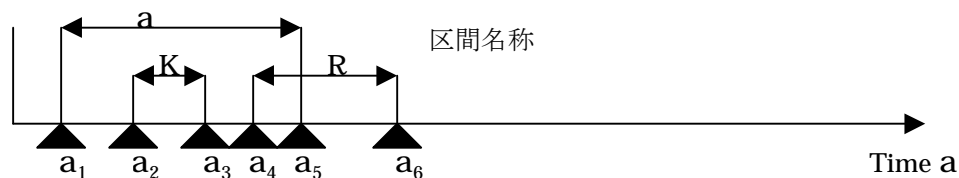
例1）

区間名称 A,S,E が上図の様にそれぞれ定義してあり、Sub-sequence List として、

Subsequence List = “ SL : E A S A S E “

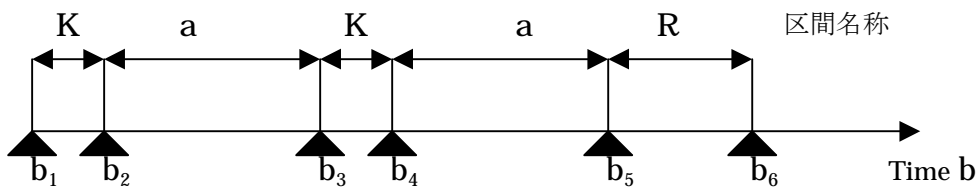
の場合、Stream Sequence としての解釈は次の図のようになる。

ここで、

$b_1 = 0$, $b_2 = a_4 - a_3$, $b_3 = b_2 + (a_2 - a_1)$, $b_4 = b_3 + (a_3 - a_2)$,
$b_5 = b_4 + (a_2 - a_1)$, $b_6 = b_5 + (a_3 - a_2)$, $b_7 = b_6 + (a_4 - a_3)$

である。

例2）

区間名称 a,K,R が上図の様にそれぞれ定義してあり、Sub-sequence List として、

Subsequence List = “ SL : K a K a R “

の場合、Stream Sequence としての解釈は次の図のようになる。

ここで、

$b_1 = 0$, $b_2 = a_3 - a_2$, $b_3 = b_2 + (a_5 - a_1)$, $b_4 = b_3 + (a_3 - a_2)$,
$b_5 = b_4 + (a_5 - a_1)$, $b_6 = b_5 + (a_6 - a_4)$

である。

(*) Note Message と Sub-sequence の Start Point と Stop Point の関係

Start Point 及び Stop Point と同様である。
Stream Sequence の最後の Phrase の Stop Point 以降に Note Message の gatetime がある場合は gatetime の発音を最後まで行う。

6.) Setup Data Chunk

音源部分の音色データやエフェクト設定等を格納する。これらのデータは多くの場合、Sequence Data Chunk 内に記述することも可能であるが、別 Data Chunk 化することでデータ・ハンドリングの便宜を計る。

Chunk ID "Mtsu" :Score Track Setup Data

body には Sequence Data Chunk で使用する Exclusive Message の並びを格納する。

Setup Data Chunk body = (Exclusive Message)+

Sequence Data Chunk で定義する Duration は含まない。理想モデルとしてこのセットアップ時間はゼロとする。セットアップ時のメッセージ間隔やメッセージ解釈時間等は System 依存であり、実装上の問題である。

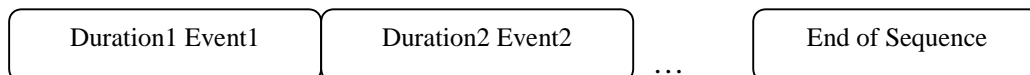
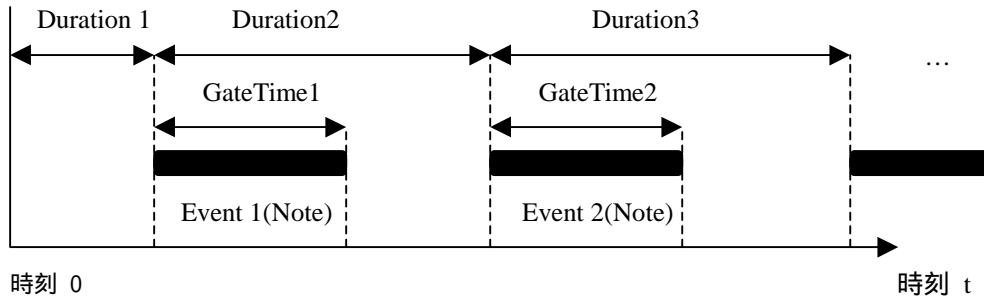
7.) Sequence Data Chunk

実演奏データを格納する。Track Chunk がある以上、この Sequence Data Chunk は必須である。

Chunk ID "Mtsq" :Score Track Sequence Data

Sequence Data Chunk は、Duration と Event をペアとしたデータ列、および End of Sequence で構成する。

Sequence Data Chunk body = ((Duration Event) | (End of Sequence))+

Duration1 Event1	Duration2 Event2	…	End of Sequence

Duration とは、1つ前の Event の開始時刻からペアとして持つ Event の開始時刻までの時間とする。例として Event が Note Message の場合、Duration と Note Message の持つ GateTime((3) b)項参照) との関係を下図に示す。Event が同一時刻では Duration は 0（0x00）である。

1つのSequence Data Chunkで可能なChannel は4 Channel までである。5 Channel 以上は複数のScore Track Chunk の Sequence Data Chunk により表現する。
各 Channel が Monophonic であるか、Polyphonic であるかは本 Format では規定しない。System でどのように解釈や処理を行うかは、個別のデータ制作ガイドライン及び実装ガイドラインなどを参照のこと。

a) End of Sequence(EOS)

連続の¥x00 4つを End of Sequence とし、Sequence の終端であることを示す。EOS は必須ではないが、Sequence の終端を示すことを明示するために記述することが望ましい。

End of Sequence = ¥x00 ¥x00 ¥x00 ¥x00

EOS は終端を意味するため、実装側の処理として、Sequence を時刻順に処理を行い EOS を検出した時点で Sequence を終了させ発音をミュートさせることが望ましい。また、Sub-sequence List を用いた場合も、Stream Sequence への展開を行った上で時刻順に処理を行い EOS を検出した時点を Sequence の終端とみなす。EOS が存在しない場合は、Chunk Size により終端位置を算出する。

b) Duration

1 byte または 2 byte で表現する。第 1byte の MSB が 0 の時は 1 byte、1 のときは 2 byte での表現になる。第 2 byte が存在する場合、MSB は常にゼロである。

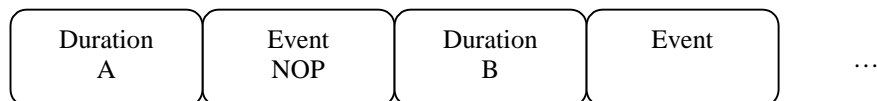
Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	S	Data1						
Data#1	0	Data2						

第 1 byte の MSB(=S)によって、表現可能な値の範囲は以下の通りである。

S	Step
0x0	0～127
0x1	128～16511

S が 1 のときは、data1 を MSB 側、data2 を LSB 側とする 14bit のデータに 128 を加えたものが表現する値となる。

1 step の分解能は、Score Track Chunk 内部の Timebase_D で設定する。1つの Duration に対し NOP Event を用いて2つ以上に分割してもよい。この際、分割した2つの Duration の和が元の Duration と等価であることとする。Step 数が 16511 を越える場合は、この方法を利用する。

Ex) Dutation が 16511＋αの場合

Duration A	Event NOP	Duration B	Event	…

A + B = 16511＋ α

c) Event

以下にEvent を定義する。使用頻度の高いExpression/ Pitch Bend/ Modulationには標準タイプと短縮タイプがあり、それぞれを場合に応じて使い分けることでデータ量を削減できる。

- Note Message

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	Channel		Octave		Note Number			
Data #1	Gatetime (1 byte 目)							
Data #2	Gatetime (2 byte 目) [オプション]							

設定したチャンネルの発音を行う。Gatetime 表現方法は Duration と同じである。ただし、GateTime = 0 は禁止とする。

Channel	Description
0x0	Channel #0
0x1	Channel #1
0x2	Channel #2
0x3	Channel #3

Octave	Description
0x0	Low
0x1	Mid Low
0x2	Mid High
0x3	High

同時に表現できるはキースケールは4オクターブまでである。
この範囲を超える音程は Control Change の Octave Shift で表現する。

Voice Number	Description
0x0	禁止
0x1	C#
0x2	D
0x3	D#
0x4	E
0x5	F
0x6	F#
0x7	G
0x8	G#
0x9	A
0xA	A#
0xB	B
0xC	C
0xD - 0xF	禁止

Mid High の A を 440(Hz) とする。

- Program change

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x0			
Data #2	Value							

該当するチャンネルの音色を設定する。音色ナンバーと音色の対応を APPENDEX 標準音色マップに従い定義する。(Bank 0x00 と 0x80 のみ)

Value	Description
0x00～0x7F	Program Change Number
0x80～0xFF	Reserved

本メッセージが指定されない場合、該当チャンネルの音色ナンバーは 0x00 とする。

- Bank Select

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x1			
Data #2	Value							

該当するチャンネルのバンクを設定する。本メッセージによりバンクを指定し、その後 Program Change を受信することで該当バンクの音色を設定する。Bank 0x00 と 0x80 のみ、APPENDEX 標準音色マップに従い音色を定義する。。

Value	Description
0x00～0x7F	Bank Number(Normal)
0x80～0xFF	Bank Number(Drum)

本メッセージが指定されない場合、該当チャンネルのバンクナンバーは 0x00 とする。

- Octave Shift

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x2			
Data #2	Value							

該当チャンネルのオクターブ・シフト量を設定する。シフト量を決定する値は絶対値であるものとする。（前のシフト値に加算しない）

Value	Description
0x00	No Shift(Original)
0x01	+1 Octave
0x02	+2 Octave
0x03	+3 Octave
0x04	+4 Octave
0x05～0x80	Reserved
0x81	-1 Octave
0x82	-2 Octave
0x83	-3 Octave
0x84	-4 Octave
0x85～0xFF	Reserved

- Modulation(標準タイプ)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x3			
Data #2	Value							

該当するチャンネルのビブラートの深さを変化させる。

Value	Description
0x00～0x7F	Data(default 0x00)
0x80～0xFF	Reserved

- Modulation(短縮タイプ)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	0	Value			

該当するチャンネルのビブラートの深さを変化させる。

Value	Description
0x1～0xE	Data(default 0x1)
0x0,0xF	Reserved

Modulation における標準タイプと短縮タイプの Data の関係は下表の通りである。

短縮タイプ	標準タイプ
0x1	0x00
0x2	0x08
0x3	0x10
0x4	0x18
0x5	0x20
0x6	0x28
0x7	0x30

短縮タイプ	標準タイプ
0x8	0x38
0x9	0x40
0xA	0x48
0xB	0x50
0xC	0x60
0xD	0x70
0xE	0x7F

- Pitch Bend（標準タイプ）

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x4			
Data #2	Value							

該当するチャンネルのピッチを上下に変化させる。

Value	Description
0x00～0x7F	Data(default 0x40)
0x80～0xFF	Reserved

- Pitch Bend(短縮タイプ)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		0	1	Value			

該当するチャンネルのピッチを上下に変化させる。

Value	Description
0x1～0xE	Data(default 0x8)
0x0,0xF	Reserved

Pitch Bend における標準タイプと短縮タイプの Data の関係は下表の通りである。

短縮タイプ	標準タイプ
0x1	0x08
0x2	0x10
0x3	0x18
0x4	0x20
0x5	0x28
0x6	0x30
0x7	0x38

短縮タイプ	標準タイプ
0x8	0x40
0x9	0x48
0xA	0x50
0xB	0x58
0xC	0x60
0xD	0x68
0xE	0x70

- Volume

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x7			
Data #2	Value							

該当するチャンネルの音量を変化させる。

Value	Description
0x00～0x7F	Data
0x80～0xFF	Reserved

以下に推奨計算式を示す。

Gain[dB] = MUTE　　　　　　, Data = 0

Gain[dB] = 20*log(Data²/127²)　, Data > 0

- Pan

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0xA			
Data #2	Value							

該当するチャンネルのステレオ音場位置を設定する。チャンネルの音を、L 側(0x01)から R 側(0x7F)間の任意の位置へ定位する。0x00 は L 側とする。

Value	Description
0x00～0x7F	Data(default 0x40)
0x80～0xFF	Reserved

以下に推奨計算式を示す。

Left　Channel Gain[dB] = 20*log(cos(π/2*Data/127))

Right Channel Gain[dB] = 20*log(sin(π/2*Data/127))

- Expression（標準タイプ）

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0xB			
Data #2	Value							

Volume で設定したチャンネルの曲中での音量を変化させる。

Value	Description
0x00～0x7F	Data(default 0x7F)
0x7F～0xFF	Reserved

以下に推奨計算式を示す。

Gain[dB] = MUTE　　　　　　　　, Data = 0

Gain[dB] = 20*log($Data^2/127^2$)　　　, Data > 0

- Expression（短縮タイプ）

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		0	0	Value			

該当するチャンネルの曲中での音量を変化させる。

Value	Description
0x1～0xE	Data(default 0xE)
0x0,0xF	Reserved

Expression における短縮タイプの Data 0x1～0x0E は、標準タイプの Data 0x00～0x7F に対し、下表の通りに対応する。

短縮タイプ	標準タイプ
0x1	0x00
0x2	0x1F
0x3	0x27
0x4	0x2F
0x5	0x37
0x6	0x3F
0x7	0x47

短縮タイプ	標準タイプ
0x8	0x4F
0x9	0x57
0xA	0x5F
0xB	0x67
0xC	0x6F
0xD	0x77
0xE	0x7F

- Exclusive Message

この Message により、デバイスに特化した各種パラメータ設定を行うことができる。
この Message は可変長である。データ形式は以下のように表記する。

	b7	b6	b5	b4	b3	b2	b1	b0
Status	1	1	1	1	1	1	1	1
Data #0	1	1	1	1	0	0	0	0
Data #1	Message Size							
Data #2	Maker ID							
Data #3	Format ID							
:	:							
Data End	1	1	1	1	0	1	1	1

Message Size は、Data_#2 から Data_End までの Byte Count とする。
Maker ID により、各社個別の Data format を Data_#3 以降に規定する。
Format ID により、各社のデバイス(機種)に特化した Data format を Data_#4 以降に規定する。

- NOP

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xFF							
Data #1	0x00							

何もオペレーションを行わない。
Data #1 が 0x00 及び 0xF0 以外は Reserved とする。

4.4.2. Mobile Standard Format（FormatType=0x01~02）

MIDI データに近い表現レベルまでサポートすることを可能としたデータ・フォーマットである。
※可変長表記について

1～4 byte の可変長で表現する。第 1～3 byte の MSB が 0 の時は以降にデータが存在しないことを示し、MSB が 1 のときは以降にデータが存在することを示す。第 4 byte の MSB は常にゼロである。

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	S	Data0						
Data#1	S	Data1						
Data#2	S	Data2						
Data#3	0	Data3						

実際の数値(Hex)	可変長での表現(Hex)
00000000	00
00000040	40
0000007F	7F
00000080	81 00
00002000	C0 00
00003FFF	FF 7F
00004000	81 80 00
00100000	C0 80 00
001FFFFF	FF FF 7F
00200000	81 80 80 00
08000000	C0 80 80 00
0FFFFFFF	FF FF FF 7F

表 可変長形式の表記例

1.)　Sequence Type

Handy Phone Standard Format に同じ。

2.)　TimeBase_D, TimeBase_G

Handy Phone Standard Format に同じ。

3.)　Channel Status

16 Channel ある Sequence data それぞれの Channel の status 情報を格納する。

Count	Channel	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	KCS		VS	LED	Reserved		Ch Type	
Data #1	2	KCS		VS	LED	Reserved		Ch Type	
Data #2	3	KCS		VS	LED	Reserved		Ch Type	
Data #3	4	KCS		VS	LED	Reserved		Ch Type	
Data #4	5	KCS		VS	LED	Reserved		Ch Type	
Data #5	6	KCS		VS	LED	Reserved		Ch Type	
Data #6	7	KCS		VS	LED	Reserved		Ch Type	
Data #7	8	KCS		VS	LED	Reserved		Ch Type	
Data #8	9	KCS		VS	LED	Reserved		Ch Type	
Data #9	10	KCS		VS	LED	Reserved		Ch Type	
Data #10	11	KCS		VS	LED	Reserved		Ch Type	
Data #11	12	KCS		VS	LED	Reserved		Ch Type	
Data #12	13	KCS		VS	LED	Reserved		Ch Type	
Data #13	14	KCS		VS	LED	Reserved		Ch Type	
Data #14	15	KCS		VS	LED	Reserved		Ch Type	
Data #15	16	KCS		VS	LED	Reserved		Ch Type	

- Key Control Status(KCS)

Key Control のリクエストを受けた時、該当 Channel に対し、Key Control を行うか否かの指定をする。ON で Key Control を有効とする。

KCS	Description
0x0	無指定
0x1	OFF
0x2	ON
0x3	Reserved

無指定における解釈は実装依存とする。

- Vibration Status(VS)

Vibration Control のリクエストを受けた時、該当 Channel のデータに同期して Vibration を行うか否かの指定をする。ON で Vibration を有効とする。

VS	Description
0x0	OFF
0x1	ON

- LED Status(LED)

LED Control のリクエストを受けた時、該当 Channel のデータに同期して LED の制御を行う否かの指定をする。ON で LED を有効とする。

LED	Description
0x0	OFF
0x1	ON

- Ch Type

該当 Channel に対し、Channel Type を指定する。

Ch Type	Description
0x0	No Care
0x1	Melody
0x2	No Melody
0x3	Rhythm

4.) Seek & Phrase Info Chunk

Handy Phone Standard Format に同じ。

5.) Setup Data Chunk

Chunk ID "Mtsu" :Score Track Setup Data

Setup Data Chunk は、Exclusive Event 列で構成する。

Setup Data Chunk body = (Exclusive Event)+
<Exclusive Event> = F0 <Length> <Data Bytes> F7

Length は、可変長形式であるとする。また最後の F7 は Length に含める。本仕様で指定されていない Exclusive Event は、本フォーマットを解釈するプログラムを作成する場合、無視する必要がある。

6.) Sequence Data Chunk

Chunk ID	"Mtsq"	:Score Track Sequence Data

Sequence Data Chunk の Body は、Duration と Event をペアとしたデータ列で構成する。Duration とは、1つ前の Event の開始時刻からペアとして持つ Event の開始時刻までの時間とする。ハフマン符号化のフォーマットは c) を参照のこと。ハフマン符号化による圧縮を行うかどうかは任意である。ただし、Format Type により下記の通りに定める。

Format Type が 0x01(Compress)の場合
Sequence Data Chunk body = Huffman(((Duration Event)|(End of Sequence))+)

Format Type が 0x02(No Compress)の場合
Sequence Data Chunk body = ((Duration Event)|(End of Sequence))+

a) Duration

可変長表記で記述する。

b) Event

status byte とは、各イベントの先頭バイトに置かれ、MSB="1"の場合を示す。

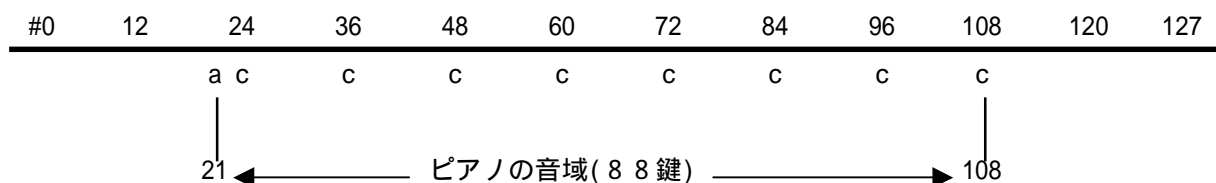
Status Byte	description
0x80~8F	Note Messege(Velocity 無)
0x90~9F	Note Messege(Velocity 有)
0xA0~AF	Reserved (Status Byte を入れて3byte)
0xB0~BF	Control Change
0xC0~CF	Program Change
0xD0~DF	Reserved (Status Byte を入れて 2byte)
0xE0~EF	Pitch Bend
0xF0	System Exclusive
0xF1~FE	Reserved
0xFF	EOS or NOP

● Note Message

Velocity 無

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	0	0	0	Channel			
Data #1	0	Note Number						
Data#2～5	S	Gatetime(Variable)						

設定 Channel に対し、Note Number のキーで発音を行う。Velocity は default で 64 とする。Gatetime は可変長表記で記述する。Note Message(velocity 有)で発音した時の velocity の値を Channel 単位で記憶し、Note Message(velocity 無)の発音では記憶した velocity の値で発音を行う。この記憶はリセット・オール・コントロールによりクリアされる(default 値となる)。

Channel は 1～16 チャンネル(0～15)、また各鍵は Note Number として割り当てられ、note on／note off messege を解釈する。Note Number 範囲は 0～127 をとり、88鍵ピアノの中央Cを Note Number =60 と定めることができる。(Note Number =69 を A=440Hz とする)

Velocity 有

Data Count	B7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	0	0	1	Channel			
Data #1	0	Note Number						
Data #2	0	Key Velocity						
Data#3～6	S	Gatetime(Variable)						

設定 Channel に対し、Note Number のキーを Key Velocity の強弱で発音を行う。Gatetime は可変長表記で記述する。Channel は 1～16 チャンネル(0～15)、Note Number は 0～127(69 を A=440Hz とする)、Key Velocity0～127 の値をとることができる。

Stream PCM

Note Messege を Stream Wave Data Chunk に格納されている Wave Data の発音指示として用いることが出来る。

1つの Note Messege に対し個々の Stream Wave Data Chunk の Wave Number へアサインする。Control Change Messege, Program Change Messege, Note Number 等のアサイン方法は、実装系に依存する。

- Control Change Message

Data Count	b7	b6	B5	b4	b3	b2	b1	b0
Data #0	1	0	1	1	Channel			
Data #1	0	Control Number						
Data #2	0	Control Value						

設定 Channel に対し、Control Number で規定されたコントローラのモディファイを行う。

- Program Change Message

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	1	0	0	Channel			
Data #1	0	Program Number						

指定 Channel の音色設定を行う。
該当するチャンネルがノーマル・チャンネルに設定されている場合、バンク・セレクトによって指定されたバンクから音色を選択する。該当するチャンネルがドラムに設定されている場合、ドラムセットを選択する。

- Pitch Bend Message

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	1	1	0	Channel			
Data #1	0	Pitch Bend Change LSB						
Data #2	0	Pitch Bend Change MSB						

該当チャンネルのピッチを上下に変化させる。変化幅（ピッチ・ベンド・レンジ）の初期値は±2 半音である。0x00/0x00 で下方向へのピッチ・ベンドが最大となる。0x7F/0x7F で上方向のピッチ・ベンドが最大となる。ピッチ・ベンド・レンジは RPN の 0x00/0x00 で設定できる。

- Exclusive Message

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xF0							
Data #x	S	Size(Variable)						
Data #y	0	Data(Variable)						
Data #end	1	1	1	1	0	1	1	1

曲中にイベントとして存在するイクスクルーシブ・メッセージを設定する。Size,Data は可変長表記で記述する。Size は Data#y から Data#end までとする。

- NOP

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xFF							
Data #1	0x00							

何もオペレーションを行わない。

- End of Sequence(EOS)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xFF							
Data #1	0x2F							
Data #2	0x00							

Sequence の終端であることを示す。EOS は必須ではないが、Sequence の終端を示すことを明示するために記述することが望ましい。(Handy Phone Standard Format の場合と異なり、Mobile Standard Format での End of Sequence は Event 扱いなので、Duration が必要。)

EOS は終端を意味するため、実装側の処理として、Sequence を時刻順に処理を行い EOS を検出した時点で Sequence を終了させ発音をミュートさせることが望ましい。また、Sub-sequence List を用いた場合も、Stream Sequence への展開を行った上で時刻順に処理を行い EOS を検出した時点を Sequence の終端とみなす。EOS が存在しない場合は、Chunk Size により終端位置を算出する。

c) Huffman 符号化

Huffman 符号化のデータフォーマットは以下とする。

Huffman(Data) = (Size) (Huffman Table) (Data)+
(Huffman Table) = ((Node/Leaf)(Code))+

Size は4バイトで、符号化する前のバイト数を表す。
ハフマンテーブルの Node/Leaf は、ビットが 0 で Leaf(葉)とし、ビットが 1 で Node(節)として Leaf の場合 0x00～0xFF までの Code を置く。Data は、ハフマンテーブルに従った可変長ビットで配置する。

7.) Stream PCM Data Chunk

Chunk ID "Mtsp" : Score Track Stream PCM data

Stream PCM 再生用の波形データを格納する。1つ以上の Stream wave Data chunk から構成される。

Stream PCM Data Chunk Body = (Stream wave Data chunk)+

a) Stream Wave Data Chunk

Chunk ID "Mwa*" : Score Track Stream Wave Data
* = 0x00 ~ 0xFF は Chunk Number である。

Chunk Number は、Wave Number を意味する。

Wave Number	Description
Ox00	Prohibition
0x01～0x3E	Wave ID
0x3F～0xFF	Prohibition

Body 内部表現は各 Wave Format に依存する。

Stream Wave Data Chunk は、先頭に波形データに対する Wave Type を付ける。その後に Wave Data を格納する。この Wave Data の内部表現は Wave Type に依存する。

Stream Wave Data Chunk body = Wave Type (Wave Data)+

以下に Wave Type のフォーマットを定義する。

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	M/S	Format			Base Bit			
Data #1	Sampling Freq(MSB)							
Data #2	Sampling Freq.(LSB)							

Channel	Description
0x0	Mono
0x1	Stereo

Format	Description
0x0	2's complement PCM
0x1	Offset Binary PCM
0x2	ADPCM(YAMAHA)

Base Bit	Description
0x0	4 bit
0x1	8 bit
0x2	12 bit
0x3	16 bit
0x4～0x7	Reserved

Sampling Freq には、波形を再生する周波数を Hz で指定する。

4.5. PCM Audio Track Chunk

Sound Track の一種類として、PCM Audio 音源用のトラック Chunk を表現する。フォーマット認識用のデータと複数の Sub Chunk を格納する。

- Format Type :1 byte（必須）
- Sequence Type :1 byte（必須）
- Wave Type :2 byte（必須）
- TimeBase_D :1 byte（必須）
- TimeBase_G :1 byte（必須）
- Seek & Phrase Info Chunk :n byte（Option）
- Setup Data Chunk :n byte（Option）
- Sequence Data Chunk :n byte（必須）
- Wave Data Chunk :n byte（必須）

1.) Format Type

このステータスでこの Track Chunk の実フォーマットを定義する。データサイズ削減のため、LSI Native Format での記述や、将来パワフルな Control CPU を想定してその他のシーケンス・フォーマットを記述可能とする。

Format Type	Description
0x00	Handy Phone Standard
0x01～0xFF	Reserved

2.) Sequence Type

Sequence Data の記述には2種類を想定する。

- 0x00 :Stream Sequence
  Sequence Data は1つの連続したシーケンス・データである。Seek Point や Phrase List はシーケンス中の意味のある位置を外部から参照する目的で利用する。

- 0x01 :Sub-sequence
  Sequence Data は複数のフレーズ・データを連続で表記したものである。Phrase List は外部から個別フレーズを認識する為に用いる。

- 0x02 ~ 0xFF（reserved）

3.) Wave Type

Wave Data Chunk のフォーマットを定義する。

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	M/S	Format			Sampling Freq			
Data #1	Base bit				Reserved			

Channel	Description
0x0	Mono
0x1	Stereo

Format	Description
0x0	Signed(2's complement) PCM
0x1	ADPCM
0x2	TwinVQ
0x3	MP3
0x4～0x7	Reserved

Sampling Freq	Description
0x0	4 kHz
0x1	8 kHz
0x2	11 kHz
0x3	22.05 kHz
0x4	44.1 kHz
0x5～0x7	Reserved

Base Bit	Description
0x0	4 bit
0x1	8 bit
0x2	12 bit
0x3	16 bit
0x4～0x7	Reserved

※Reserved は、エンコーダーテーブル等の認識に使用予定。

4.) TimeBase_D, TimeBase_G

Sequence Data 内部で使用する基準時間を表現する。
TimeBase_D が Duration の基準時間、TimeBase_G が GateTime の基準時間である。

TimeBase_D,G	Description
0x00	1 msec
0x01	2 msec
0x02	4 msec
0x03	5 msec
0x04～0x0F	Reserved
0x10	10 msec
0x11	20 msec
0x12	40 msec
0x13	50 msec
0x14～0xFF	Reserved

5.) Seek & Phrase Info Chunk（Option）

Sequence Data 内で意図的なポジションからスタート／ストップさせるため、又は意味のある区間リピート等を表現可能とする為の情報を格納する。この運用、及び機能はオプションであり、機種やデータを作成するときの意図に依存する。（詳細は4.4.1参照のこと）

6.) Setup Data Chunk（Option）

PCM エンコーダ部分のパラメータやエフェクト設定等を格納する。これらのデータは多くの場合、Sequence Data Chunk 内に記述することも可能であるが、別 Data Chunk 化することでデータ・ハンドリングの便宜を計る。

7.) Sequence Data Chunk

Audio Event データを格納する。Track Chunk がある以上、この Sequence Data Chunk は必須である。

8.) Wave Data Chunk

Wave データを格納する。Track Chunk がある以上、この Audio Data Chunk は最低一個必須である。Chunk ID の最後の1byte を Audio Data Number とし、最大 255 Wave が記述可能である。

4.5.1. Handy Phone Standard Format

1.) Seek & Phrase Info Chunk

Chunk ID	"AspI"	:Audio Track Seek & Phrase Info

body に以下の順でデータを格納する。

- Start Point :1 ~ 4 byte（Option）
- Stop Point :1 ~ 4 byte（Option）
- Phrase List :n byte（Option）
- Sub-sequence List :n byte（Option）

内容、及び用途は Score Track と同じである。

2.) Setup Data Chunk

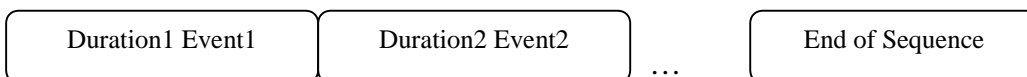
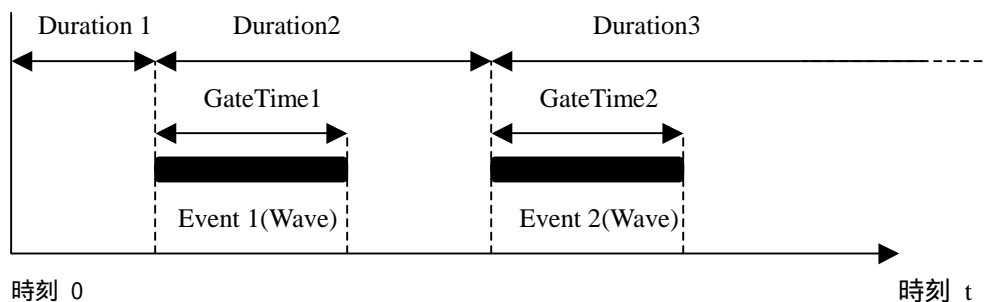
Chunk ID	"Atsu"	:Audio Track Setup Data

内容、及び用途は Score Track と同じである。

3.) Sequence Data Chunk

Chunk ID	"Atsq"	:Audio Track Sequence Data Chunk

Sequence Data Chunk は、Duration と Event をペアとしたデータ列、および End of Sequence で構成する。

Sequence Data Chunk body = ((Duration Event)|(End of Sequence))+

Duration とは、1つ前の Event の開始時刻からペアとして持つ Event の開始時刻までの時間とする。例として Event が Wave Message の場合、Duration と Note Message の持つ GateTime((4) c)項参照) との関係を下図に示す。Duration が 0（0x00）の場合、Event は同一時刻に実行される。

a) End of Sequence

連続の¥x00 4つを End of Sequence とし、Sequence の終端であることを示す。EOS は必須ではないが、Sequence の終端を示すことを明示するために記述することが望ましい。

End of Sequence = ¥x00 ¥x00 ¥x00 ¥x00

EOS は終端を意味するため、実装側の処理として、Sequence を時刻順に処理を行い EOS を検出した時点で Sequence を終了させ発音をミュートさせることが望ましい。また、Sub-sequence List を用いた場合も、Sub-sequence を時刻列へ展開を行った上で時刻順に処理を行い EOS を検出した時点を Sequence の終端とみなす。EOS が存在しない場合は、Chunk Size により終端位置を算出する。

b) Duration

Max 2 byte で表現する。

87654321	87654321	Type	Step
0xxxxxxx		Short	0～ 127
1xxxxxxx	0xxxxxxx	Medium	128～16511

1step の分解能は Seek & Phrase Info Chunk 内部の TimeBase_D で設定する。
Step 数が 16511 を越える場合は、NOP Event を挿入する。

c) Event

以下に Event を定義する。使用頻度の高い Expression/ Pitch Bend/ Modulation には標準タイプと短縮タイプがあり、それぞれを場合に応じて使い分けることでデータ量を削減できる。

● Wave Message

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	Channel		Wave Number					
Data #1	Gatetime (1 byte 目)							
Data #2	Gatetime (2 byte 目) [オプション]							

設定したチャンネルの Wave 再生を行う。Gatetime 表現方法は Duration と同じである。GateTime = 0 は Reserved とする。

Channel	Description
0x0	Channel #0
0x1	Channel #1
0x2	Channel #2
0x3	Channel #3

Wave Number	Description
0x01～0x3E	Wave ID
0x00,0x3F	Prohibition

● Pitch Bend(標準タイプ)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x4			
Data #2	Value							

該当するチャンネルのピッチを上下に変化させる。

Value	Description
0x00～0x7F	Value(default 0x40)
0x80～0xFF	Reserved

- Pitch Bend(短縮タイプ)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		0	1	Value			

該当するチャンネルのピッチを上下に変化させる。

Value	Description
0x1～0xE	Value(default 0x8)
0x0,0xF	Reserved

短縮タイプの Data 0x1～0x0E は、標準タイプの Data 0x00～0x7F に対し、0x0E ステップ分の値と対応する。Pitch Bend における標準タイプと短縮タイプの Data の関係は下表の通りである。

短縮タイプ	標準タイプ
0x1	0x08
0x2	0x10
0x3	0x18
0x4	0x20
0x5	0x28
0x6	0x30
0x7	0x38

短縮タイプ	標準タイプ
0x8	0x40
0x9	0x48
0xA	0x50
0xB	0x58
0xC	0x60
0xD	0x68
0xE	0x70

- Volume

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x7			
Data #2	Value							

該当するチャンネルの音量を変化させる。

Value	Description
0x00～0x7F	Value
0x80～0xFF	Reserved

以下に推奨計算式を示す。

Gain[dB] = MUTE　　　　　　　　, Value = 0

Gain[dB] = $20*\log(Data^2/127^2)$　　, Value > 0

- Pan

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0xA			
Data #2	Value							

該当するチャンネルのステレオ音場位置を設定する。チャンネルの音を、L 側(0x01)から R 側(0x7F)間の任意の位置へ定位する。0x00 は L 側とする。

Value	Description
0x00～0x7F	Value(default 0x40)
0x80～0xFF	Reserved

以下に推奨計算式を示す。

Left　Channel Gain[dB] = $20*\log(\cos(\pi/2*Data127))$

Right Channel Gain[dB] = $20*\log(\sin(\pi/2*Data/127))$

- Expression（標準タイプ）

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0xB			
Data #2	Value							

Volume で設定したチャンネルの曲中での音量を変化させる。

Value	Description
0x00～0x7F	Value(default 0x7F)
0x7F～0xFF	Reserved

以下に推奨計算式を示す。

Gain[dB] = MUTE , Value = 0

Gain[dB] = 20*log(Data²/127²) , Value > 0

- Expression（短縮タイプ）

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		0	0	Value			

該当するチャンネルの曲中での音量を変化させる。

Value	Description
0x1～0xE	Value(default 0xE)
0x0,0xF	Reserved

Expression における短縮タイプの Data 0x1～0x0E は、標準タイプの Data 0x00～0x7F に対し、下表の通りに対応する。

短縮タイプ	標準タイプ
0x1	0x00
0x2	0x1F
0x3	0x27
0x4	0x2F
0x5	0x37
0x6	0x3F
0x7	0x47

短縮タイプ	標準タイプ
0x8	0x4F
0x9	0x57
0xA	0x5F
0xB	0x67
0xC	0x6F
0xD	0x77
0xE	0x7F

- Exclusive Message

この Message により、デバイスに特化した各種パラメータ設定を行うことができる。
この Message は可変長である。データ形式は以下のように表記する。

	b7	b6	b5	b4	b3	b2	b1	b0
Status	1	1	1	1	1	1	1	1
Data #0	1	1	1	1	0	0	0	0
Data #1	Message Size							
Data #2	Maker ID							
Data #3	Format ID							
:	:							
Data End	1	1	1	1	0	1	1	1

Message Size は、Data_#2から Data_End までの Byte Count とする。
Maker ID により、各社個別の Data format を Data_#3 以降に規定する。
Format ID により、各社のデバイス(機種)に特化した Data format を Data_#4 以降に規定する。

- NOP

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xFF							
Data #1	0x00							

何もオペレーションを行わない。
Data #1 が 0x00 及び 0xF0 以外は Reserved とする。

4.) Wave Data Chunk

Chunk ID "Awa*" :Audio Track Wave Data Chunk
* = 0x00 ~ 0xFF は Chunk Number である。

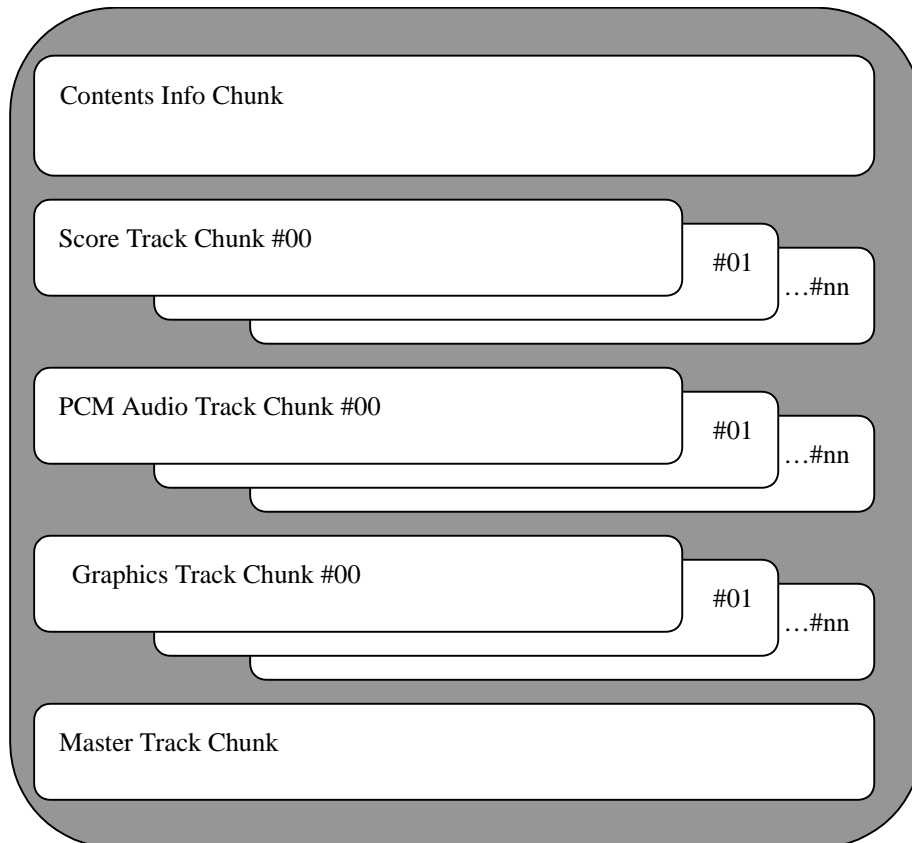
Chunk Number は、Sequence Data Chunk の Wave Message で使用する。
Wave Number を意味する。

Wave Number	Description
Ox00	Prohibition
0x01～0x3E	Wave ID
0x3F～0xFF	Prohibition

Body 内部表現は各 Wave Format に依存する。

5. Graphics Track 基本概念

5.1. Graphics Track の位置付け

Graphics Track は SMAF 仕様データの中で、表示を行うためのシーケンスを表現するデータ表現です。

SMAF Track 概念図

Graphics Track は上図のように SMAF を構成する Track の一つです。

それぞれの Track は対応する出力デバイスに対するシーケンスを定義するためのデータ表現です。
Score Track は音源デバイスに対する演奏シーケンス、
PCM Track は PCM 再生デバイスに対する発音シーケンス、
Graphics Track は表示デバイスに対する描画シーケンスを記述しています。

SMAF 仕様では、全ての Track は時刻 0 で同時にスタートすると定義します。
SMAF の再生系は、各 Track の時間表現に従って同時並行に再生処理を進めることで結果的に、
複数の出力デバイスに対する再生処理が同期して行われることになります。

データ表現上、各 Track は任意の数(上限 256Track)だけ表記することが可能ですが、SMAF 再生系の
制限や、コンテンツ制作上の制限から実用上の Track 数が決められます。以下の解説では Graphics Track
は最大1個を前提とします。これは仕様設計時点で想定している SMAF の再生系では表示デバイスは1種
類しかないからです。

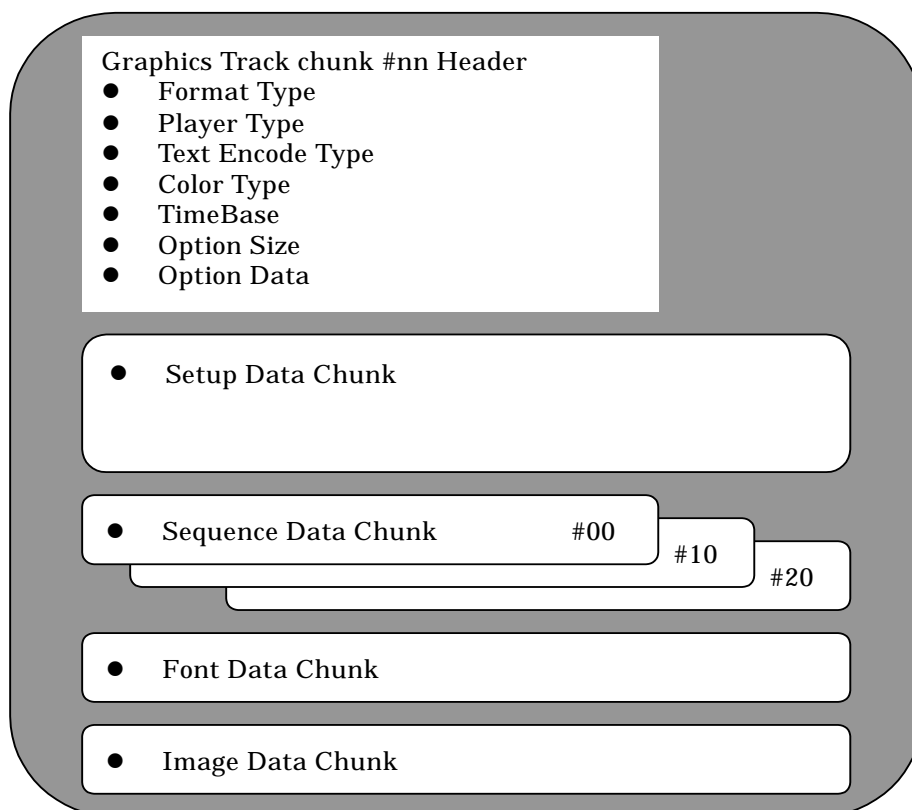
Graphics Track が想定する表示デバイスは複数の仮想プレーンを持ちます。Graphics Track には各仮想プレーンに対応した描画シーケンスを記述可能です。その描画シーケンスを Graphics Track のサブシーケンスと呼びます。各々のサブシーケンスではテキストや画像データを時間に同期して表示、消去するための手続きを定義します。

物理表示デバイス(携帯端末の LCD 表示パネルなど)には仮想プレーンを合成した結果を表示します。本解説書では仮想プレーンの合成方法についても定義しています。仮想プレーンの数はデータ表現上任意の数だけ(最大 256)定義できますが、本解説書では3プレーンでの実装を前提に解説を行います。(内1つは背景色のみが指定可能です。)

5.2. Graphics Track の基本構造

Graphics Track は

- Header (必須)
- Setup Data Chunk (必須)
- Sequence Data Chunk (必須 1個以上)
- Font Data Chunk (最大 1 個 Optional)
- Image Data Chunk (最大 1 個 Optional)

から構成されています。

Graphics Track 概念図

5.3. 仮想表示デバイス

再生系からみた表示デバイスは仮想的に 3 階層の仮想プレーンがあると仮定し、ソフトウェアにより各プレーンを合成することで最終的な表示イメージを生成します。各プレーンは物理表示デバイスと同じサイズで 256 色指定が可能とします。

仮想プレーンの定義

仮想プレーンとはメモリー領域に確保した仮想的な表示デバイスのことです。

コンテンツ制作の便宜上、各仮想プレーンを以下のように呼ぶことにします。

- Plane 0 : バックドロッププレーン(背景色指定用,描画不可)
- Plane 1 : 背景画像プレーン(主に画像表示用)
- Plane 2 : テキストプレーン(主にテキスト表示用)

仮想プレーンの数は、Graphics Track の Sequence Data Chunk の個数(サブシーケンスの数)だけ定義することができますが、本節では以上の3プレーンでの実装を前提に説明を行います。

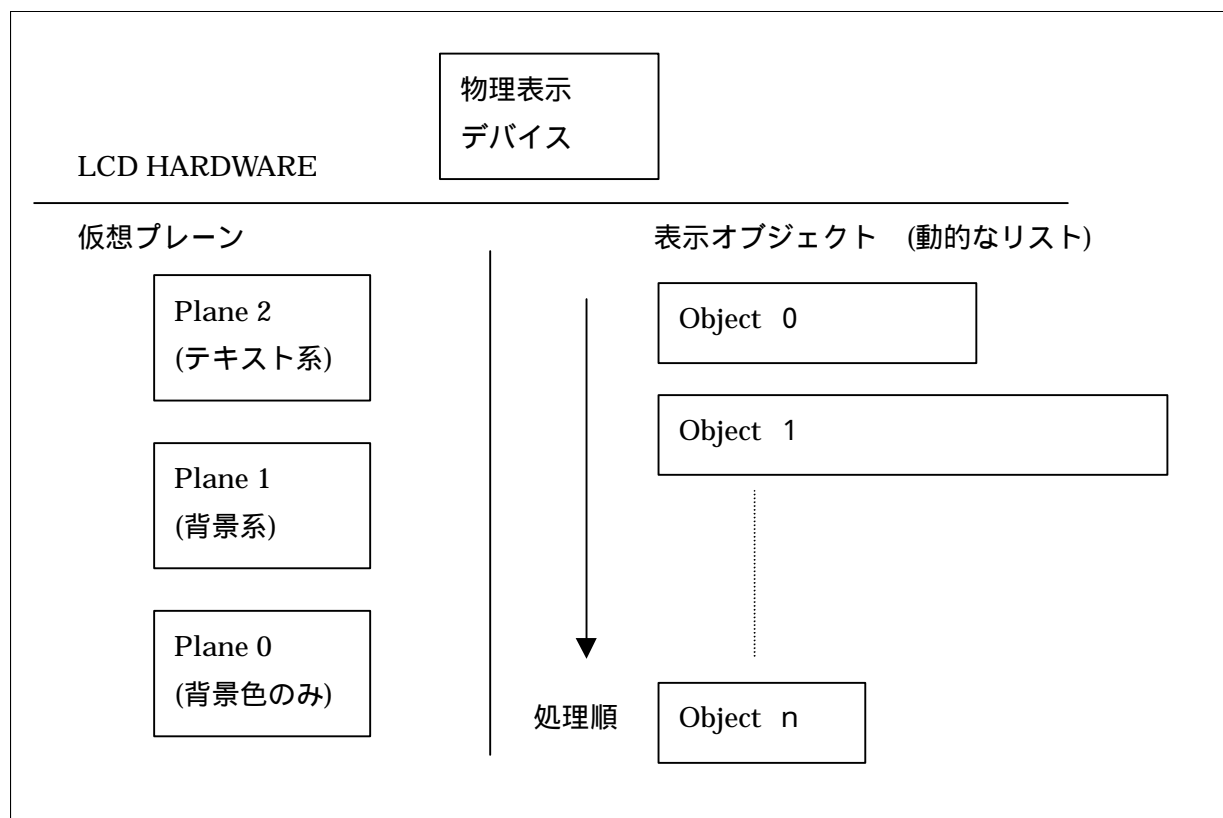
仮想プレーン合成のアルゴリズム

プレーン番号の小さいプレーンが下層にあるものとします。

上層のプレーンに表示した表示オブジェクトは下層の表示オブジェクト(後述)をオーバーラップします。

表示オブジェクトの無い領域は「透明」と解釈し、下層のプレーンが見えます。

表示オブジェクトを消去すると消去した領域は原則として「透明」になります。(表示オブジェクトが重なった場合は別途定義します。)

表示オブジェクトは矩形ですが矩形の中に透明ピクセルを定義することができます。

表示デバイスと仮想表示デバイス 概念図

5.4. Sequence Data Chunk と仮想表示デバイス

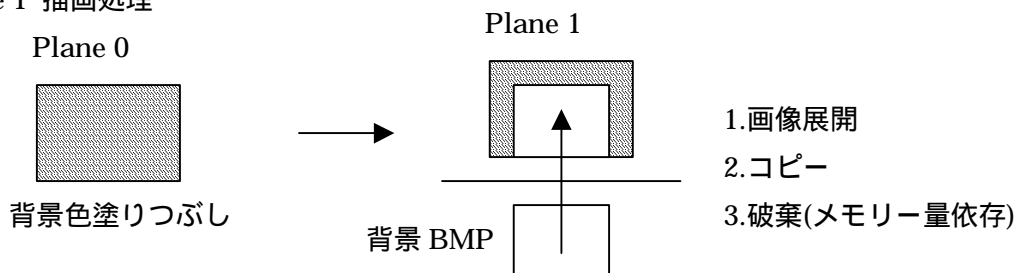
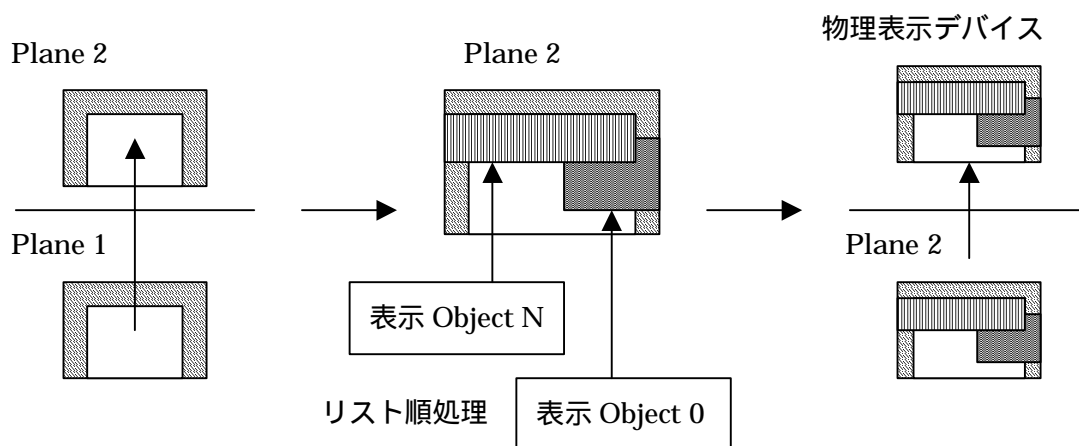
Sequence Data Chunk と仮想プレーンを関連付けは以下の通りです。

- Sequence Data 0x00 が Plane 0 に対するシーケンス (BackDrop Color のみ変更)
- Sequence Data 0x10 が Plane 1 に対するシーケンス
- Sequence Date 0x20 が Plane 2 に対するシーケンス

同じ Graphics Track 内の複数の Sequence Data は Sequence Data No の小さいものが下層にあると解釈します。

5.5. 仮想プレーンの合成アルゴリズム

表示オブジェクトは仮想プレーン上の矩形領域(表示領域)を占める画像です。
表示オブジェクトには寿命があり、寿命の終わりには必ず消去します。

Plane1と Plane2では対応するシーケンスデータに記述された表示オブジェクトがリストとして管理されます。

標準的な描画処理順序 概念図

Plane 0 の場合

仮想プレーン領域は不要で、Plane 1 を背景色で塗り潰します。

Plane 1 とPlane 2 の場合

後から書いた方が先に書いた表示オブジェクトの前面に表示されます。

上位のオブジェクトを消去すると、下位のオブジェクトが見えるようになります。

表示オブジェクトの表示期間中表示内容を保存する必要があります。表示オブジェクトは描画履歴順のリストで管理し、表示更新の度に表示内容を再合成する必要があります。

表示オブジェクトは同一プレーン内を移動させることができます。

移動は表示オブジェクトの消去->再表示の繰り返しです。

移動に伴い他の表示オブジェクトと表示領域が重複した場合の動作は、上記の説明に準じます。

Plane 1,2 とも標準的な描画処理を行えば元あった状態に戻ります。

表示オブジェクトに AuxiliarySub-block によるサブシーケンス(§6.3.6)を付属させることで表示オブ

ジェクトの移動や効果を表現します。

5.6. 座標系

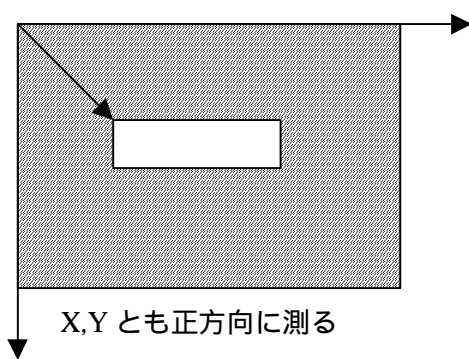
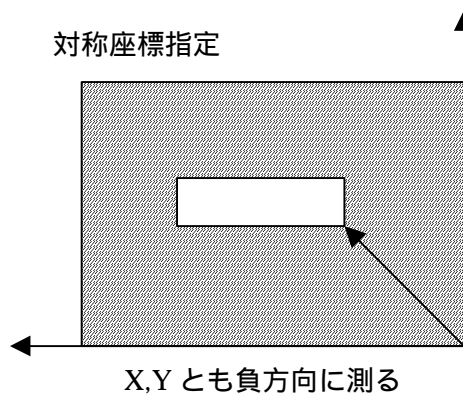
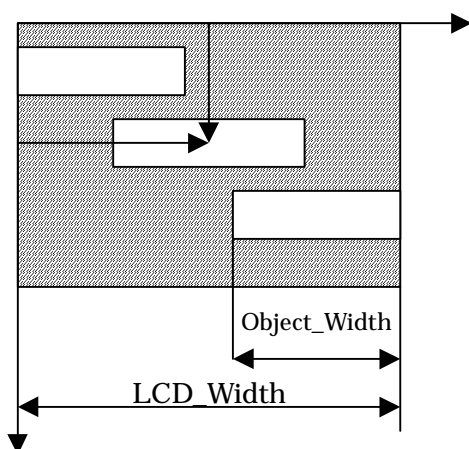
物理デバイスや仮想プレーン上の位置を表現するための座標系と座標指定の種類について説明します。
また表示オブジェクト内部のピクセルの位置を指定するためのローカル座標系について説明します。

5.6.1. 仮想プレーン上の表示座標指定

SMAF ファイル内のシーケンスデータには、仮想プレーン上の表示オブジェクトの表示位置が記述されています。
表示オブジェクトの表示位置の基準座標は以下のいずれかの方法で表現することができます。

- 標準座標指定
  座標原点は左上。X 軸は右が正方向。Y 軸は下が正方向です。
  標準座標指定では表示オブジェクトの左上の座標を指定します。

- 対称座標指定
  表示画面右端または下端から負の方向に図った座標です。
  座標原点は右下。X 軸は左が正方向。Y 軸は上が正方向です。
  対称座標指定では表示オブジェクトの右下の座標を指定します。

- レイアウト情報指定
  表示座標を計算する方法を指定する方式
    100 分率で指定。
    X 方向には 0%:左詰、50%センタリング,100%:右詰
    Y 方向には 0%:上詰め、50%:センタリング,100%:下詰め

レイアウト情報指定

100 分率で指定する。
X 方向には　0：左詰
　　　　　　50：センタリング
　　　　　100：右詰

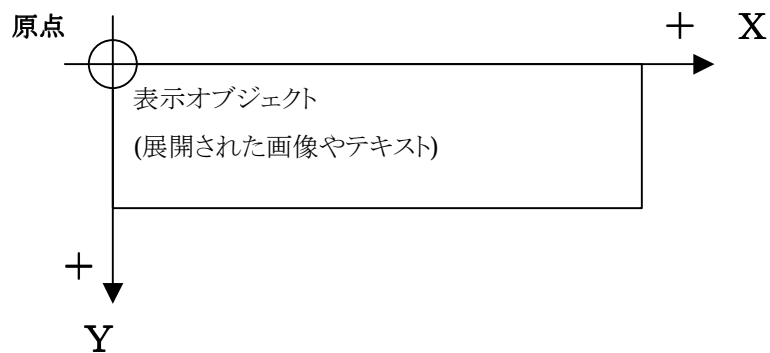
Y 方向には　0：上詰
　　　　　　50：センタリング
　　　　　100：下詰

標準座標系 指定値 ＝ (LCD_Width － Object_Width) ＊ Layout_Ratio ／ 100
LCD_Width < Object Width の場合、表示オブジェクトは
0　　　では、右側が
50　　　では、両側が
100　　では、左側が画面上からはみ出します。

- 混在指定

X 座標、Y 座標それぞれ独立に、座標指定をするか、レイアウト情報指定をするかを選択できるようにします。

5.6.2. 表示オブジェクトのローカル座標系

表示オブジェクト上のピクセルの位置は表示オブジェクトの左上を原点として X 方向右向き正、Y 方向下向き正の座標系で指定します。

6. Graphics Track データフォーマット

6.1. Graphics Track Chunk

Track Chunk ID においては、先頭 3byte でどの Track であるかを表わし、最後の1byte で Track Number を Binary 値で表現します。
以下にGraphics Track Chunk の Chunk ID を記します。

Chunk ID "GTR*" :Graphics Track * = 0x00 ~ 0xFF

Graphics Track の内部構成は以下の通りです。

- Header (必須)
- Setup Data Chunk (必須)
- Sequence Data Chunk (1個以上、必須)
- Font Data Chunk (最大1つ、Optional)
- Image Data Chunk (最大1つ、Optional)

Header の内容

- Format Type :1byte
- Player Type :1byte
- Text Encode Type :1byte
- Color Type :1byte
- TimeBase :1byte
- Option Size :1byte
- Option Data :Option Size で指定したサイズ(0～255b)

1.) Format Type

使用するグラフィックデータのデータフォーマットを定義します。

Format Type	Description
0x00	Handy Phone Standard
0x01～0xFF	Reserved

2.) Player Type

想定する再生環境のクラスを定義します。

Player Type	Description
0x00	Handy Phone Standard
0x01～0xFF	Reserved

3.)　Text Encode Type

使用する文字コード体系を定義します。

Contents Code Type	Description	Language
0x00	Shift-JIS	日本語
0x01	Latin-1	英語、フランス語、ドイツ語 イタリア語、スペイン語、ポルトガル語
0x02	EUC-KR(KS)	韓国語
0x03	HZ-GB-2312	中国語(簡体字)
0x04	Big5	中国語(繁体字)
0x05	KOI8-R	ロシア語など
0x06	TCVN-5773:1993	ベトナム語
0x07～0x1F	Reserved	Reserved
0x20	UCS-2	Unicode
0x21	UCS-4	Unicode
0x22	UTF-7	Unicode
0x23	UTF-8	Unicode
0x24	UTF-16	Unicode
0x25	UTF-32	Unicode
0x26～0xFF	Reserved	Reserved

4.)　Color Type

使用するカラーコード体系を定義します。

Color Type	Description
0x00	Direct RGB:=3:3:2
0x01	Index Color
0x02～FF	Reserved

SMAF データ中のテキストや背景などの色指定は 256 色カラー表示のみ対応します。
色指定は 0 から 255 までの数値で行います。
SMAF 内に格納された画像データ自身が持っている色表現は 256 色カラーに制限されません。
ただし SMAF 再生系は画像展開時に 256 色に減色して使用することになります。

5.) TimeBase

内部で使用する基準時間を定義します。

TimeBase が Duration、LifeTime などの Graphics Track 内の基準時間となります。

時間を表すデータに TimeBase を掛けることで実時間を計算します。

TimeBase	Description
0x00	1 msec
0x01	2 msec
0x02	4 msec
0x03	5 msec
0x04～0x0F	Reserved
0x10	10 msec
0x11	20 msec
0x12	40 msec
0x13	50 msec
0x14	60 msec
0x15	70 msec
0x16	80 msec
0x17	90 msec
0x18	100 msec
0x19～0xFF	Reserved

6.) Option Size

Option に続く拡張用データの Size を指定します。

7.) Option Data

拡張用データを Option Size 分(0～255b)指定します。

6.2. Setup Data Chunk

Setup Data Chunk は表示に必要な各種パラメータを定義するのに使用します。（1 個必須）

Chunk ID "Gtsu" :Graphics Track Setup Data

Setup Data Chunk の内容

Display Parameter Definition Chunk (1 個必須)

Color Palette Definition Chunk (Optional)

6.2.1. Display Parameter Definition Chunk

Display Parameter Definition Chunk は表示 Event Type 毎に固有の表示パラメータを定義するのに使用します。

Chunk ID "Gdpd" :Display Parameter Definition Chunk

Default の表示パラメータを定義するのが必須です。

Display Object Event Type ごとのパラメータ定義は Option です。

データ構造は可変長レコードの配列で、EventType 毎の Definition Data の繰り返しです。

Display Parameter Definition Chunk

```
EventType 毎の繰り返し {
    RecordSize      :1～2byte   各 Event Type ごとのデータ長 Duration 表現を採用します。
    Event Type      :1byte      以下の表示パラメータを使用する Event Type の指定
    繰り返し {
        PrmID       :1byte
        Value       :1byte
    }
}
```

Event Type = 0x00 を指定して Default Parameter を定義します。

この Event Type は Setup Data Chunk 内でしか使用しません。

有効な Event Type は 0x40...0x7f の範囲です（後述）。それ以外の Event Type に対する定義は無視します。

PrmID	Description	
0x01	Font Type	フォントの種類
0x02	Font Size	フォントのサイズ
0x03	Direction	文字並び方向
0x04	Attribute	文字アトリビュート指定 将来拡張
0x05～0x0F	Reserved	
0x10	Font Color 0	文字色
0x11	Font Color 1	色替え後文字色
0x12	Edge Color 0	文字縁取り色 将来拡張
0x13	Edge Color 1	色替え後文字縁取り色 将来拡張
0x14	Back Color 0	文字背景色
0x15	Back Color 1	色替え後文字背景色
0x16～0x1F	Reserved	
0x20	Coordinates	デフォルトの座標指定方法
0x21～0x2F	Reserved	
0x30	BackDropColor	背景色 Plane 0 指定色
0x31	Transparent Color	透明色とする色を指定
0x32	Transparent Enable	透明処理の有効フラグ
0x16～0xFF	Reserved	

文字アトリビュート指定(Bold,Italic,Outline,Shadow) Font 指定

Coordinates

指定方法は§6.3.2を参照のこと

BackDropColor

Setup Definition Chunk 内の Type = 0x00 指定によるデフォルト設定でのみ有効とします。

その他 Plane 0 以外のシーケンスデータにおける BackDropColor の変更は無視します。

Transparent Enable

0x00 のとき透明処理を無効、それ以外は有効とします。

6.2.2. Color Palette Definition Chunk

Chunk ID "Gcpd" :Color Palette Definition Chunk

Color Palette Definition Chunk は詳細は現在未定義

Header の Color Type が 0x01 で Color Palette Definition Chunk が無い場合は

システムのカラーパレットに依存します。

6.3. Graphics Track Sequence Data Chunk

6.3.1. Graphics Track Sequence Data Chunk

Graphics Track Sequence Data Chunk は表示シーケンスを表現します。

Chunk ID	"Gsq*"	:Graphics Track Sequence Data	* = 0x00 ～ 0xFF

Sequence Data Chunk の ChunkID は先頭 3byte でどの Chunk であるかを表し、最後の 1 byte で Sequence Data Number を Binary 値で表現します。

Sequence Data Number = *	Description
0x00	Plane 0
0x01～0x0F	Reserved
0x10	Plane 1
0x11～0x1F	Reserved
0x20	Plane 2
0x21～0xFF	Reserved

再生系は表示データを合成するための仮想表示プレーンを複数持ち、その Plane と Sequence Data を関連付けて使用します。Number が小さい方が下層にあると定義します。

Sequence Data 表現

Sequence Data は可変長バイトストリームです。

Sequence Data の構成要素は Duration と Event と EOS です。

Sequence Data の先頭要素は Duration とし、Duration と Event は必ず交互に表記します。

EOS は Sequence の終了(End Of Sequence)の意味。4バイト以上の 0x00 の連続で表現します。

Sequence Data の解釈にあたっては、Event,Duration の解釈に先立って EOS の確認をします。
0x00 が 4 バイト以上連続する Event や Duration が存在しないことをデータフォーマット
として保証します。
Duration = 0x00 と NOP Event = 0x00 の連続によって、4バイト以上 0x00 が連続した
場合は EOS と解釈します。以後の Sequence Data があった場合は Size 情報を手がかり
に読み飛ばします。

Sequence Data を表現するための数値表現は Coordinates で用いる表現と Duration で用いる
表現があります。
SMAF 中で座標指定するための Coordinates 内の数値表現と、時間や長さなどの指定に使用する
Duration 表現を以下の節で説明します。

「Coordval 表現を採用します。」という記述をしている数値は、指定の方法として Coordval の 1～2 バイトの

表現を使用します。

同様に「Duration 表現を採用します。」という記述をしている数値は、指定の方法として Duration の 1～2 バイトの表現を使用します。

6.3.2. Coordinates と Coordval

座標表現詳細(Coordinates)

Coordinates := [Format 指定+] Position+Position

Coordinates	b7	b6	b5	b4	b3	b2	b1	b0
Format 指定	1	1	1	1	xx		yy	
Position	X 座標の 1byte 目							
[Option]	X 座標の 2byte 目 [Option]							
Position	Y 座標の 1byte 目							
[Option]	Y 座標の 2byte 目 [Option]							

一つ目の Position　が X 座標、二つ目が Y 座標を表します。

Format 指定:= '1111xxyy'

xx が X 座標のフォーマット

yy が Y 座標のフォーマット

xx 又は yy	Description
0x0	標準座標指定
0x1	対称座標指定
0x2	レイアウト指定
0x3	指定無し

Format 指定は省略できる場合があります。省略しない場合は指定の通り解釈します。

Format 指定を省略した場合はデフォルト指定を採用します。

初期値は X,Y 共に標準座標指定。

省略された場合の座標系指定の解釈順序について

1. Display Parameter Definition Chunk に Type = 00 でデフォルトの座標指定 Format を定義します。
2. またその指定をオーバーライドするように各 Event Type 毎の座標指定 Format を定義することも可能です。
3. さらに Sequence Data Chunk 内の Event においても Parameter Override Sub-block を追加して Display Parameter Definition Chunk の Type 毎の Format 指定をオーバーライドする設定ができます。

同一の Event(Primary から Auxiliary の最後まで)の中で座標指定を含む場合は以下のような解釈の特例を設けます。

Event 中(Primary から Auxiliary の最後まで)で Data 毎の個別 Format 指定を行った
場合上の3つの方法による指定をオーバーライドするデフォルト値として採用し、
次の個別 Format 指定までか、他に指定がなければ Auxiliary Sub-block の
最後までそのデフォルト指定を有効とします。

Position := 標準座標指定 | 対称座標指定 | レイアウト情報指定

標準座標指定 :1～2byte := Coordval 表現(後述)を採用します。

対称座標指定 :1～2byte := Coordval 表現(後述)を採用します。

レイアウト情報 :1byte := 0...127 で 0...100%を表現

Coordval:=座標値を表現する可変長データ表現(1～2byte で表記)

表現可能範囲は-2048 から+2047 まで。

1バイト表記(0～223 を表現)

0=>0x00

1=>0x01

...

222=>0xDE

223=>0xDF

2バイト表記(-2048 から 2047 までを表現)

0xExxx の xxx を 12 ビットの符号付き整数と解釈します。

-2048=>0xE800

-2047=>0xE801

...

-1 => 0xEFFF

0 => 0xE000

1 => 0xE001

...

2047=>0xE7FF

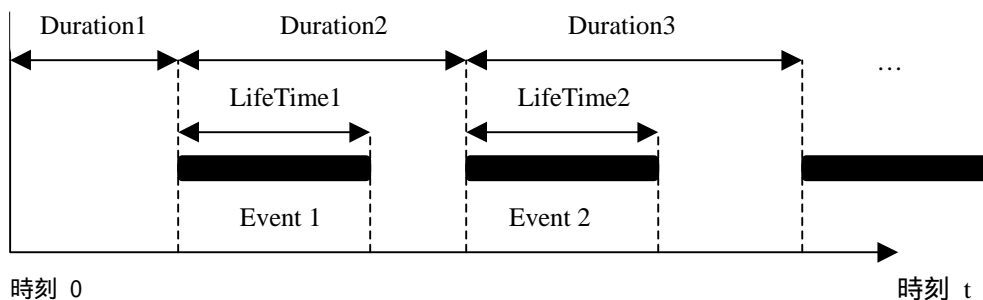
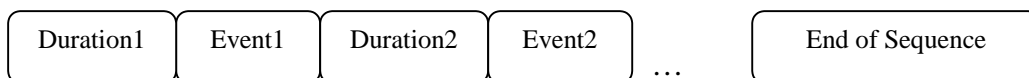
6.3.3. Duration

Duration の表現

Duration は Event 間の時間間隔を指定しています。

Duration は「TimeBase で指定した時間」を単位として表現しています。

Duration 値をデータの先頭から累積することにより、直後の Event の開始時刻を決定できます。

Event 開始時刻 ＝(直前までの Duration の総和) ＊ TimeBase

Sequence Data 概念図

1 byte または 2 byte で表現します。第 1byte の MSB が 0 の時は 1 byte での表現で、1 のときは 2 byte での表現になります。第 2 byte が存在する場合、第2byte の MSB は常にゼロです。

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#1	S	Data1						
Data#2	0	Data2						

第 1 byte の MSB(=S)によって、表現可能な値の範囲は以下の通りです。

S	Step
0x0	0～127
0x1	128～16511

S が 1 のときは、data1 を MSB 側、data2 を LSB 側とする 14bit のデータに 128 を加えたものが表現する値となります。

1 バイト表現

0x00～0x7f := 0x00 から 0x7f（0～127）

2 バイト表現

0x80,0x00 := 0x0080(128)

0x80,0x01 := 0x0081(129)

0x81,0x00 := 0x0100(256)

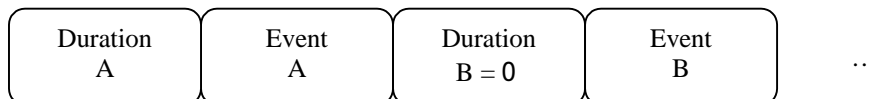
...

0xff,0x7f := 0x407f(16511)

1 step の分解能は、Graphics Track Chunk 内部の TimeBase で設定します。1 つの Duration に対し NOP Event を用いて 2 つ以上に分割してもかまいません。この際、分割した 2 つの Duration の和が元の Duration と等価であることとします。Step 数が 16511 を越える場合は、この方法を利用します。

Ex)必要な Dutation が 16511＋α = A + B の場合

同時刻に起こる複数の Event を定義するためには例えば Event A と Event B の間に Duration = 0 を挿入します。

Ex) Event A と B を同時刻に定義する。

6.3.4. 終端定義

Graphics Track の長さは、Graphics Track に含まれるサブシーケンス(Sequence Data Chunk)の中の最長のサブシーケンスの長さ、と定義します。

サブシーケンスの長さは EOS(End Of Sequence)の位置と規定します。ただし、Graphics Track に含まれる複数のサブシーケンスの長さがそれぞれ異なっていた場合、SMAF の再生系は一番長いサブシーケンスの EOS と同じ時刻に全てのサブシーケンスの EOS があるものと解釈することとします。

サブシーケンスに記述した Display Object Event の消去時刻(表示時刻+LifeTime)が EOS を超過した場合は、EOS の位置を優先します。ここで言う EOS は前述の「補正された EOS」を指します。EOS のタイミングで表示中の Display Object があったら強制的に消去すると同時にそのサブシーケンスは終了します。つまり最後の Event についてその表示オブジェクトの LifeTime 期間の表示を保証するために LifeTime 以上の Duration を挿入しておく必要があります。

Display Object Event に含まれる Auxiliary Sub-block に記述したシーケンスデータが LifeTime より長い場合も考えられます。この場合も上位階層の定義、つまり LifeTime の長さを優先して解釈するものとします。Display Object Event の消去時刻(表示時刻+LifeTime)の時点で実行中の Auxiliary Sub-block のシーケンスは強制的に中断すると同時に Display Object は消去されます。

本来、Auxiliary Sub-block に記述したシーケンスデータの長さは LifeTime 以下であるのが望ましいし、Display Object Event の消去時刻(表示時刻+LifeTime)は EOS と同じか前に位置するようにデータを作成すべきです。以上は、この規定を守らないデータを解釈するときの再生系の挙動を規定するための説明です。

同様に、Graphics Track と Score Track PCM Track 内の各 Sequence Data Chunk に記述されたシーケンスデータの長さに差がある場合も、もっとも長い EOS を採用することとします。

例えば
Score Track のシーケンスデータが Graphics Track のシーケンスデータよりも長い場合は、
Graphics の最後の表示オブジェクトの消去又は EOS の解釈による消去後は背景色のみが表示され、
最も長い EOS の終了まで維持されます。

6.3.5. Event

Event の標準形式の定義

Event Type	:1byte	
Event Size	:1～2byte	Duration 表現を採用します。
Event Data	:Event Size bytes	0...16511byte 可変長

Event Size は Event Data のサイズを示します。

Eventには

Short Control Event

Control Event

Display Object Event

3種類があります。

Event の詳細

1.) Short Control Event

(1 バイトで表現します。Event Size および EventData はありません。)

Event Type = 0x00..0x1f

Event Type	Description
0x00	NOP
0x01	Reset Origin Event
0x02～0x1F	Reserved

- NOP Event (No OPration Event 1 バイトで表現。Event Size、Event Data 無し)

  Event Type = 0x00 は 1 バイトの Event で NOP と解釈し、何もしません。

- Reset Origin Event

  Offset Origin Event でオフセットされた原点を、オフセットされていない元の指定にもどします。

  オフセットされていない場合は、何もしません。

  Reset Origin Event の置かれた Graphics Sequence Track のみに有効です。

2.) Control Event

Event Type = 0x20..0x3f

Event Size	:1～2byte	Duration 表現を採用します。
Event Data	: Event Size bytes	0....16511byte 可変長

Event Type	Description
0x20	BackDrop Color Definition Event
0x21	Offset Origin Event
0x22	User Event
0x23～0x3F	Reserved

- BackDrop Color Definition Event

	b7	b6	b5	b4	b3	b2	b1	b0
Event Type	0x20							
Event Size	0x01							
Event Data	BackDrop Color							

背景色(BackDrop Color)を定義します。Plane ０（Chunk ID = Gsq0)以外の指定は無視されます。

Value	Description
0x00～0xFF	BackDrop Color

- Offset Origin Event

	Description	b7	b6	b5	b4	b3	b2	b1	b0
Event Type		0x21							
Event Size		0x05~0x09							
Event Data	Coordinates	1	1	1	1	xx		yy	
		X 座標の 1byte 目							
		X 座標の 2byte 目 [Option]							
		Y 座標の 1byte 目							
		Y 座標の 2byte 目 [Option]							
	Xwidth	Xwidth の 1byte 目							
		Xwidth の 2byte 目 [Option]							
	Ywidth	Ywidth の 1byte 目							
		Ywidth の 2byte 目 [Option]							

Event Type = 0x21

Event Size :1～2byte Duration 表現を採用します。

Event Data

Coordinates :3～5byte オフセット用矩形の配置座標を指定します。
座標系の指定を省略不可とし必須とします。

Xwidth :1～2byte Duration 表現を採用します。

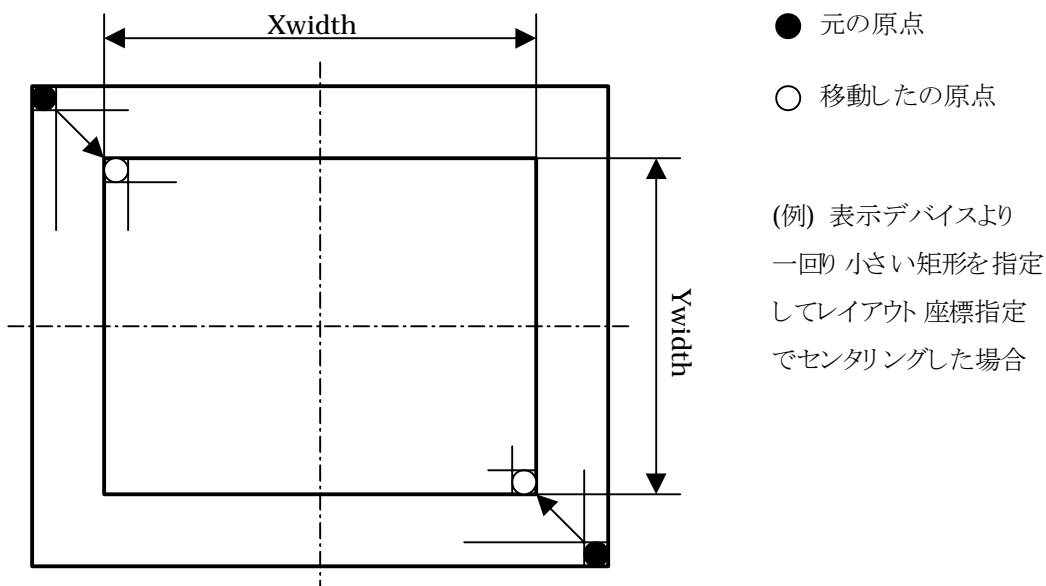
Ywidth :1～2byte Duration 表現を採用します。

標準座標指定の原点(画面左上) および 対象座標指定の原点(画面右下) の位置を移動します。
レイアウト座標指定の計算には影響しません。

Xwidth,Ywidth の指定が 0x00 の場合は 0x01 を指定したものと解釈します。

この Control Event が指定された以降の Event における標準座標指定と対称座標指定の原点が移動するものとします。次にこの Control Event が出現するか、無ければシーケンスの終端まで有効であるとします。

表示オブジェクトの表示中に変更された場合は、その表示オブジェクトの表示 Event の解釈時点の原点の状態を優先します。つまり LifeTime による解釈よりも Event の前後関係に依存するものとします。

Reset Origin Event の有効範囲は置かれた Graphics Sequence Track だけで他の Sequence Track に影響しません。

原点が移動するのは標準座標系および対象座標系指定です。
レイアウト座標指定における左詰めや右詰めは元のサイズのまま計算され、
この Event の矩形領域に影響を受けません。

- User Event

	b7	b6	b5	b4	b3	b2	b1	b0
Event Type	0x22							
Event Size	0x01							
Event Data	0	0	0	0	User Event ID			

Event Data は 0x00～0x0F からユーザーが任意に選択します。
User Event は Plane0だけに置かれ、Plane1 や Plane2 に置くことはできません。

3.) Display Object Event

- 汎用表示 Event

Event Type	Description
0x40～0x7F	汎用表示 Event
0x80～0xFF	Reserved

<table>
<tr><td></td><td></td><td>b7</td><td>b6</td><td>b5</td><td>b4</td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr>
<tr><td>Event Type</td><td></td><td colspan="8">0x40 ～ 0x7F</td></tr>
<tr><td rowspan="2">Event Size</td><td></td><td colspan="8">Size の 1byte 目</td></tr>
<tr><td></td><td colspan="8">Size の 2byte 目 [Option]</td></tr>
<tr><td rowspan="2">LifeTime</td><td></td><td colspan="8">LifeTime の 1byte 目</td></tr>
<tr><td></td><td colspan="8">LifeTime の 2byte 目 [Option]</td></tr>
<tr><td rowspan="5">Coordinates</td><td>省略可</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">xx</td><td colspan="2">yy</td></tr>
<tr><td></td><td colspan="8">X 座標の 1byte 目</td></tr>
<tr><td></td><td colspan="8">X 座標の 2byte 目 [Option]</td></tr>
<tr><td></td><td colspan="8">Y 座標の 1byte 目</td></tr>
<tr><td></td><td colspan="8">Y 座標の 2byte 目 [Option]</td></tr>
<tr><td>Sub-block</td><td>Primary 必須</td><td colspan="8">別記 (可変長)</td></tr>
<tr><td></td><td>Auxiliary</td><td colspan="8">別記 (可変長) [Option]</td></tr>
</table>

汎用表示 Event は Primary Object Sub-block が必須です。

Event Type = 0x40...0x7F　識別子として使用します。
Event Size　:1～2byte　Duration 表現を採用します。
Event Data
　LifeTime　:1～2byte　Duration 表現を採用し、表示開始から消去までの時間を指定します。
　Coordinates　:2～5byte　表示位置の座標指定
　Object Sub-block　:可変長 Option 表示オブジェクトの指定や動作バリエーションを指定。

Event Type は Setup Data Chunk 内の Display Parameter Definition Chunk で定義している Event Type 毎の表示パラメータを取り出し、表示処理に反映させるのに使うための識別子です。
SMAF データ定義上は汎用表示 Event に Event Type 毎の定義の差は存在しません。
Display Parameter Definition Chunk(§6.2.1)ではその意味付けを実現する色などのパラメータをセットすることになります。

Event Size は直後の LifeTime から最後の Object Sub-block の終端までのデータサイズをバイト単位で表します。

LifeTime は Sub-block で指定した Display Object を表示開始から消去するまでの期間を表しています。
LifeTime は TimeBase を単位として指定します。
Coordinetes は表示オブジェクトの表示位置を指定します。

表示オブジェクトの内容は Display Object Event の中の Object Sub-block の中で定義します。

SMAF 仕様としては汎用表示 Event の個々の Event Type に意味づけを行いません。
データ作成のガイドラインに規則として各 Event Type の運用上の意味付けを行います。
例) 歌詞、男声パート歌詞、女声パート歌詞、混声パート歌詞
曲タイトル、作詞者名、作曲者名、歌手名
セリフ、合いの手、コメント,etc...

6.3.6. Object Sub-block

Object Sub-block データの一般形式は以下の通りです。

Sub-block Type :1byte

Sub-block Size :1～2byte

Sub-block Body :Sub-block Size bytes

Object Sub-block には Primary と Auxiliary の2種類があります。

Primary Sub-block はEventで用いられる表示オブジェクトの中味を指定します。

Auxiliary Sub-block は表示オブジェクトにつける動作の修飾部分(効果や変化)を指定します。

1.) Primary Sub-block

Primary Sub-block は表示オブジェクトの内容を記述します。

汎用表示 Event に対しては Primary Sub-block を 1 個だけ必ず指定します。

Primary Sub-block には以下の 7 種類があります。

- Text (テキスト)
- Bitmap (2 値画像)
- Image (任意画像)
- Rectangle (矩形領域)
- Text Block (改行付きテキスト)
- Image Tile (画像タイル)
- Bitmap Tile (2 値画像タイル)

Sub-block Type	Description
0x01	Text
0x02	Bitmap
0x03	Image
0x04	Rectangle
0x05	Text Block
0x06	Image Tile
0x07	Bitmap Tile
0x08～7F	Reserved

- Text 表示オブジェクト

表示オブジェクトとして 1 行分の文字列を指定します。

Graphics Track Chunk の Header で指定した文字コードの文字列 (可変長)

0x00 は End Of Data(EOD)と解釈。文字列が切れているとみなされます。

EOD は省略可とします。 EOD は無い場合は Size Data から終端を判断します。
文字コードによっては EOD が正しく解釈できないため、その場合 EOD は使用しません。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x01							
Size	Size の 1byte 目							
	Size の 2byte 目 [Option]							
Body	文字列データ(可変長)							

Sub-block Body :可変長

指定された文字列データに対して、Event Type から決定した表示パラメータを作用させて
カラー化した画像を生成します。

文字列データには「外字」を埋め込むことが可能です。Text Encode Type が S-JIS 指定の場合
JIS 区点コードの 14 区、15 区を外字コード領域と解釈します。また Unicode の場合には
Private Use Area を外字コードと解釈します。これらの範囲の文字コードが出現したら
Font Data Chunk(§6.4.1)に含まれる外字ビットマップデータを文字フォントデータとして使用し文字表示処理を行います。
外字データで指定されたコードが Font Data Chunk に存在しなかった場合はその Font Size
毎のスペースを表示します。

文字列が無い場合には表示 Event を無視します。

- Bitmap 表示オブジェクト

表示オブジェクトとして二値画像を指定します。Bmp Chunk 内のデータを指定します。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x02							
Size	0x01							
Body	画像番号							

Sub-block Body :1byte := 画像番号

Primary Sub-block Body で画像番号を指定します。画像番号は Image Data Chunk 内の
Bmp Chunk(§6.5.2)の ChunkID の 4 バイト目の数値です。
指定された Bitmap 画像データに対して EventType から決定した表示パラメータを作用させて
カラー化した画像を生成します。
Bitmap 画像データは展開時の 1 を文字表示色 0 を文字背景色として表示します。

したがって Bitmap(2 値画像)を指定することで、任意の形状の文字列を定義したかのように

振る舞わせることが可能です。

指定された画像番号が無い場合は表示 Event を無視します。

- Image 表示オブジェクト

表示オブジェクトとして画像データを指定します。Image Chunk 内のデータを指定します。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x03							
Size	0x01							
Body	画像番号							

Sub-block Body :1byte := 画像番号

Primary Sub-block Body で画像番号を指定します。画像番号は Image Data Chunk 内の Image Chunk(§6.5.1)の ChunkID の 4 バイト目の数値です。
指定された画像データを展開して SMAF 表示オブジェクトにする際に256色に合わせる必要があります。

指定された画像番号が無い場合は表示 Event を無視します。

- Rectangle(矩形)表示オブジェクト

矩形領域と表示色を指定して矩形を塗り潰した表示オブジェクトを指定します。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x04							
Size	0x03～0x05							
Body	Xwidth の 1byte 目							
	Xwidth の 2byte 目 [Option]							
	Ywidth の 1byte 目							
	Ywidth の 2byte 目 [Option]							
	Color							

Sub-block Body :=
X 方向 dot 数 Xwidth :1～2byte Duration 表現を採用します。
Y 方向 dot 数 Ywidth :1～2byte Duration 表現を採用します。
塗り潰し Color :1byte

Xwidth 又は Ywidth の少なくとも一方が 0x00 の場合は表示 Event を無視します。

- TextBlock 表示オブジェクト

表示オブジェクトとして改行付きのテキストを描画します。

テキスト解釈や表示の基本的な部分は Text 表示オブジェクトに準じます。

Graphics Track Chunk の Header で指定した文字コードの文字列 (可変長) 。
文字列は Size Data から終端を判断します。
文字コードによっては EOD が解釈できないため、EOD は使用しません。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x05							
Size	Size の 1byte 目							
	Size の 2byte 目 [Option]							
Body	Block Width の 1byte 目							
	Block Width の 2byte 目[Option]							
	Line Space の 1byte 目							
	Line Space の 2byte 目 [Option]							
	BackColor							
	文字列データ(可変長)							

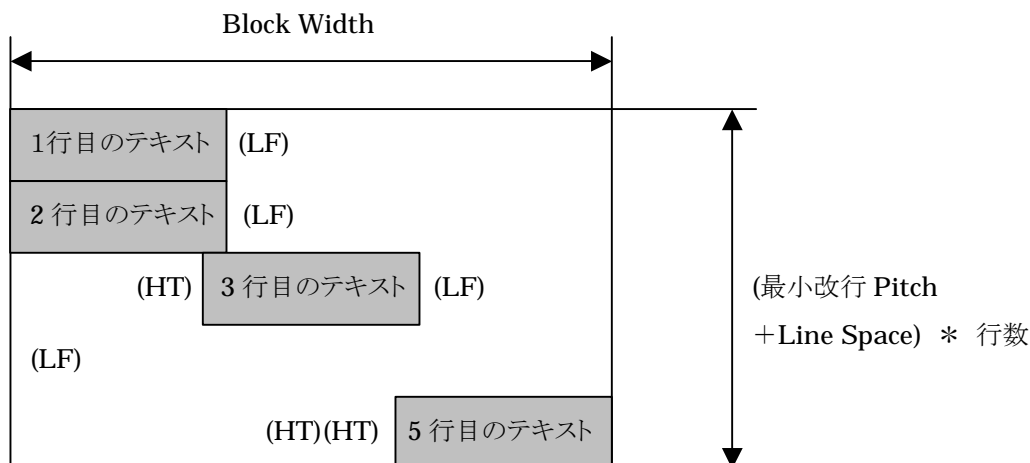
Sub-block Body　　:可変長

Block Width	:1～2byte	Coordval 表現を採用します。 (横書きの場合巾 縦書きの場合高さ)
Line Space	:1～2byte	Duration 表現を採用します。
BackColor	:1byte	ブロック内の背景色を定義します。
Text	:可変長	

Block Width は 12bit 符号付整数(Coordval 表現)として扱います。(－2048～2047)
0 を指定した場合システムの LCD 巾を指定したものとみなします。
正の数は実際のブロックサイズをドット数で指定します。
負の数はシステムの LCD サイズから差し引くドット数値を指定します。

Line Space は行間に挿入するドット数を指定します。
0 を指定した場合はシステムフォントサイズに依存した最小のテキスト改行ピッチを指定したものとみなします。Line Space を大きくすることによって生じたスペースは TextBlock の BackColor で塗りつぶします。

文字列データのコントロールコードとして以下を解釈します。

改行	LF	(例 S-JIS の場合 0x0A)
センタリング	HT	(例 S-JIS の場合 0x09)
右詰	HT HT	(例 S-JIS の場合 0x09 0x09 の 2byte)

改行コード無しでテキストが Block Width を越えて並んだ場合は、Block Width を基準にワードラップして表示します。強制改行は次の文字が LF でない場合に挿入されます。
Font Size(全角)より小さい Block Width は表示 Event を無視します。
縦書きの指定の場合は、Block Width は縦サイズ Line Space は横方向の行間ドット数となります。

- Image Tile 表示オブジェクト

表示オブジェクトとして Image Chunk 内の任意画像をタイル状に並べて描画します。
Primary Sub-block Body で画像番号を指定します。画像番号は Image Data Chunk 内の Image Chunk(§6.5.1)の ChunkID の 4 バイト目の数値です。
指定された画像データを展開して SMAF 表示オブジェクトにする際に256色に合わせる必要があります。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x06							
Size	0x05～0x0A							
Body	画像番号							
	Xwidth の 1byte 目							
	Xwidth の 2byte 目 [Option]							
	Ywidth の 1byte 目							
	Ywidth の 2byte 目 [Option]							
	1	1	1	1	xx		yy	
	X 座標の 1byte 目							
	X 座標の 2byte 目 [Option]							
	Y 座標の 1byte 目							
	Y 座標の 2byte 目 [Option]							

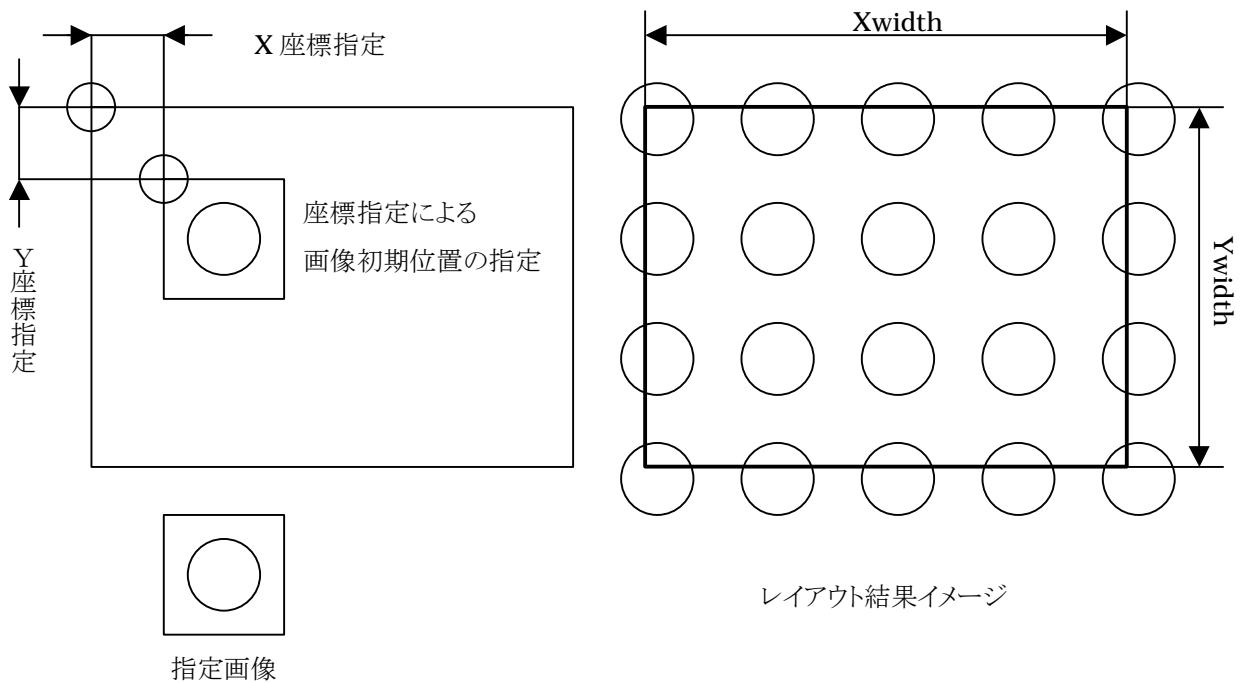
Sub-block Body

画像番号　　　　　:1byte　 Image Chunk 内の画像を指定します。

画像サイズ　　　　:Xwidth Ywidth の画像サイズを指定します。

貼り付け位相座標　:2～5byte Coordinates 表現を採用します。

Xwidth = 0
Ywidth = 0　のときは大きさをそれぞれ LCD サイズとします。

指定の画像を基準位置からタイルレイアウトを行い物理表示デバイスと同じサイズに並べます。

- Bitmap Tile 表示オブジェクト

表示オブジェクトとして Bmp Chunk 内の任意画像をタイル状に並べて描画します。
Primary Sub-block Body で画像番号を指定します。画像番号は Image Data Chunk 内の Bmp Chunk(§6.5.1)の ChunkID の 4 バイト目の数値です。
指定された画像データを展開して SMAF 表示オブジェクトにする際に Display Parameter の文字色と文字背景色を使用します。

<table>
<tr><th>Sub-block</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td>Type</td><td colspan="8">0x07</td></tr>
<tr><td>Size</td><td colspan="8">0x05～0x0A</td></tr>
<tr><td rowspan="10">Body</td><td colspan="8">画像番号</td></tr>
<tr><td colspan="8">Xwidth の 1byte 目</td></tr>
<tr><td colspan="8">Xwidth の 2byte 目 [Option]</td></tr>
<tr><td colspan="8">Ywidth の 1byte 目</td></tr>
<tr><td colspan="8">Ywidth の 2byte 目 [Option]</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">xx</td><td colspan="2">yy</td></tr>
<tr><td colspan="8">X 座標の 1byte 目</td></tr>
<tr><td colspan="8">X 座標の 2byte 目 [Option]</td></tr>
<tr><td colspan="8">Y 座標の 1byte 目</td></tr>
<tr><td colspan="8">Y 座標の 2byte 目 [Option]</td></tr>
</table>

Sub-block Body

画像番号　:1byte　Bmp Chunk 内の画像を指定します。

画像サイズ　:Xwidth Ywidth の画像サイズを指定します。

貼り付け位相座標　:2～5byte Coordinates 表現を採用します。

Xwidth = 0

Ywidth = 0　のときは大きさをそれぞれ LCD サイズとします。

指定の画像を基準位置からタイルレイアウトを行い物理表示デバイスと同じサイズに並べます。

並べ方は Image Tile の定義と同じです。

2.)　Auxiliary Sub-block

Auxiliary Sub-block は Primary Sub-block で指定した表示オブジェクトに対して様々な修飾を加えるためのデータ表現です。

Auxiliary Sub-block を含まない Display Object Event は Primary Sub-block の内容を指定期間(=LifeTime)表示した後、消去するだけです。

Auxiliary Sub-block は表示オブジェクトの表示内容を変化させたり、表示位置を移動させたり、表示領域を変更したりするサブシーケンスです。

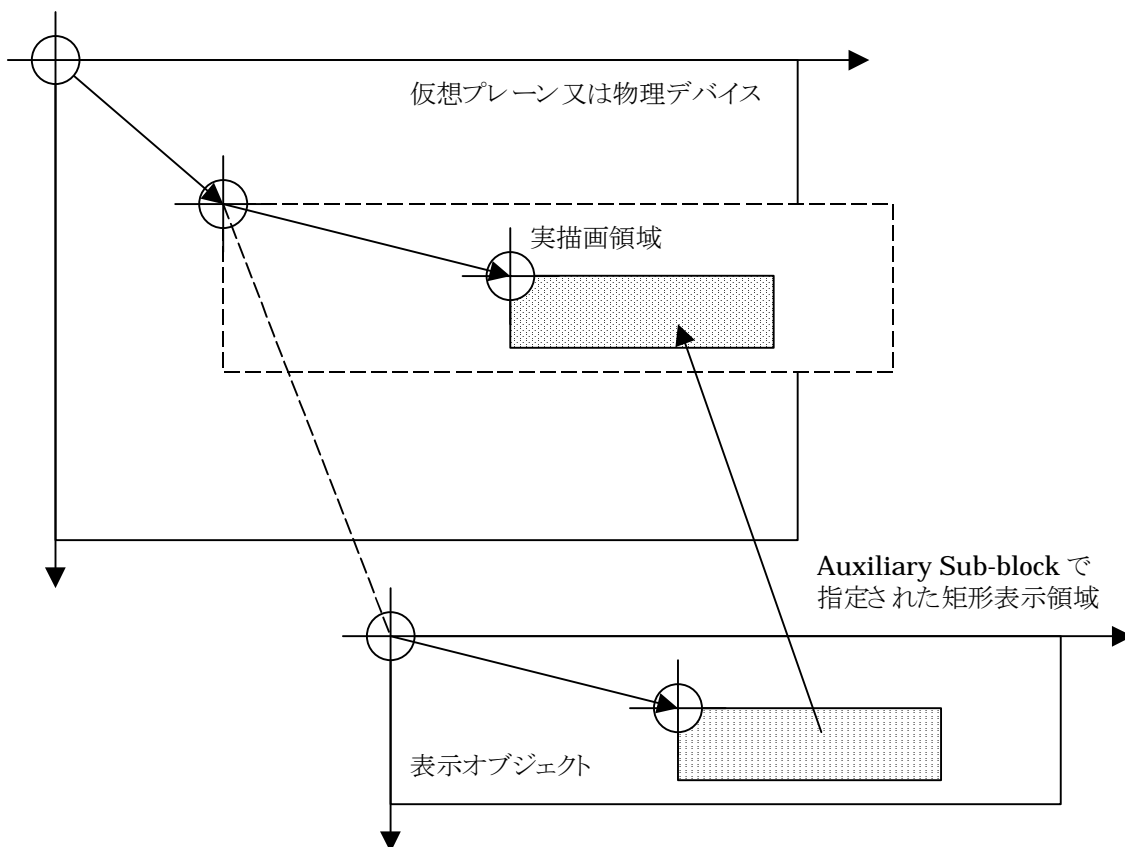
Auxiliary Sub-block には

- Display Parameter(表示パラメータ)の変更
- テキストの色替えシーケンス
- バナー表示シーケンス
- 表示位置移動シーケンス
- 点滅表示シーケンス
- Fill-InOut シーケンス

などがあります。

また複数の Sub-block を記述することで動作の組合せを表現することができます。

ただし、同じ Sub-block Type のシーケンスを複数個並列に記述することはできません。

Sub-block Type	Description
0x80	Parameter override
0x81	WipeTiming
0x82	WipeSeq
0x83	BannerInfo
0x84	TravelSeq
0x85	BlinkSeq
0x86	ColorBlinkSeq
0x87	Fill-InOutSeq
0x88～FF	Reserved

表示オブジェクトと表示領域の指定概念図

- Parameter override

Event Type 毎に指定されている表示パラメータをオーバーライドするためのデータ表現。
Display Parameter Definition Chunk に指定のある Event Type でも表示色やフォントなどの表示パラメータを変更したい時に使用します。

Sub-block Type := 0x80:Parameter override
Sub-block Body :=
繰り返し {
  prmID :1byte
  Value :1byte
}

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x80							
Size	可変長							
Body	[以下 2byte 単位の繰り返し]							
	prmID							
	Value							

PrmID	Description
0x01	Font Type
0x02	Font Size
0x03	Direction（文字並び方向）
0x04	Attribute（文字アトリビュート指定　将来拡張）
0x05～0x0F	Reserved
0x10	Font Color 0（文字色）
0x11	Font Color 1（色替え後文字色）
0x12	Edge Color 0（文字縁取り色　将来拡張）
0x13	Edge Color 1（色替え後文字縁取り色　将来拡張）
0x14	Back Color 0（文字背景色）
0x15	Back Color 1（色替え後文字背景色）
0x16～0x1F	Reserved
0x20	Coordinates (デフォルトの座標指定方法)
0x21～0x2F	Reserved
0x30	BackDropColor (背景色　Plane 0　指定色)
0x31	Transparent Color
0x32	Transparent Enable
0x16～0xFF	Reserved

- WipeTiming

表示オブジェクトの表示期間中に表示色を変更することができます。

表示オブジェクトに色替えを行う時間を指定します。

表示開始時刻から、ここで指定した時間が経過したら文字列全体を一度に色替えをします。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x81							
Size	0x01〜0x02							
Body	WipeTime の 1byte 目							
	WipeTime の 2byte 目 [Option]							

Sub-block Type := 0x81:WipeTiming

Sub-block Body :=

WipeTime　　:1〜2byte　　Duration 表現を採用します。

WipeTime は TimeBase を基準時間として指定します。

表示 Event の開始時間からの相対時間を指定します。

TextBlock に WipeTiming を指定した場合は TextBlock 内の全て背景色部分を除くテキスト部分が色替えされます。

- WipeSeq

表示オブジェクトの表示期間中に1文字単位の表示色を変更することができます。

色替えを行う時間を 1 つ以上の複数指定します。

表示開始時刻から、指定した時間が経過する度に表示オブジェクトの先頭文字から一文字ずつ順に色替えをします。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x82							
Size	可変長							
Body	Entry の 1byte 目							
	Entry の 2byte 目 [Option]							
	Timing Data の 1byte 目							
	Timing Data の 2byte 目 [Option]							
	〜							
	以下 Entry 数だけ Timing Data のくりかえし							

Sub-block Type := 0x82:WipeSeq

Sub-block Body :=

Entry　　:1〜2byte　　Duration 表現を採用します。

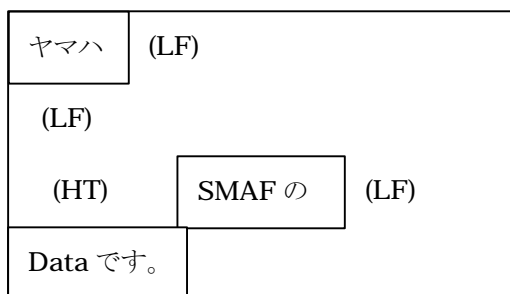
Timing Data の個数を指定します。

Timing Data　　:1～2byte　　Duration 表現を採用します。Entry 数だけくりかえします。

Timing Data は、TimeBase を基準時間として指定します。
Timing Data の各エントリは、ある文字を色替えしてから、その次の文字の色替えまでの
相対時間です。
ただし、エントリの先頭の Timing Data は、イベントの表示開始時間から、最初の文字の
色替えまでの時間です。

対応する 表示 Event は Text と TextBlock とします。
Timing Data = 0x00 の場合前の文字と同時に色替えを行います。

Entry 数と Data 数の対応
Entry=< Data 数　Entry 数の Data を解釈します。
Entry>　Data 数　Data を全て解釈し、Entry 数を減らします。
さらに
Entry 数と 表示 Event の文字数 の対応
Entry =< 文字数　対応する文字数までを色替えします。残った文字は色替えされません。
Entry >　文字数　余った Entry は無視します。
を合わせて解釈します。

TextBlock に WipeSeq を指定した場合、コントロールコードを除く実テキスト部分の
文字数を先頭から数えてその部分を該当する時間に色替えします。

No.	Data	色替え実時間	色替え文字
Entry 0	1 秒	1 秒	ヤ
Entry 1	1 秒	2 秒	マ
Entry 2	1 秒	3 秒	ハ
Entry 3	3 秒	6 秒	S
Entry 4	0 秒	6 秒	M
Entry 5	0 秒	6 秒	A
Entry 6	0 秒	6 秒	F
Entry 7	1秒	7 秒	の

コントロールコードは色替えの対象ではありません。
『 SMAF』は表示開始から6秒後に同時に色替えされます。
『 Data です。』はこの場合色替えされません。

TextBlock 色替えの例

- BannerInfo

表示オブジェクトの表示期間中に表示オブジェクトをバナー表示させることができます。

<table>
<tr><th>Sub-block</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td>Type</td><td colspan="8">0x83</td></tr>
<tr><td>Size</td><td colspan="8">0x07～0x12</td></tr>
<tr><td rowspan="11">Body</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="4">BannerType</td></tr>
<tr><td colspan="8">BackColor</td></tr>
<tr><td colspan="8">WindowSpan の 1byte 目</td></tr>
<tr><td colspan="8">WindowSpan の 2byte 目 [Option]</td></tr>
<tr><td colspan="8">EffectiveSpan の 1byte 目</td></tr>
<tr><td colspan="8">EffectiveSpan の 2byte 目 [Option]</td></tr>
<tr><td colspan="8">InitialPosition の 1byte 目</td></tr>
<tr><td colspan="8">InitialPosition の 2byte 目 [Option]</td></tr>
<tr><td colspan="8">BannerTravel の 1byte 目</td></tr>
<tr><td colspan="8">BannerTravel の 2byte 目 [Option]</td></tr>
<tr><td colspan="8">BannerTime 1byte 目</td></tr>
<tr><td></td><td colspan="8">BannerTime 2byte 目 [Option]</td></tr>
</table>

Sub-block Type := 0x83:BannarInfo.

Sub-block Body :=

BannarType :1byte

Bit3 = 0:水平バナー、1:垂直バナー

Bit2 = 0:正方向、1:負方向

画面上に向かって表示オブジェクトが右から左、下から上へ動くバナーを正方向とします。

Bit1 = 負方向(左側又は上側)の折り返し表示 0:無し、1:あり

Bit0 = 正方向(右側又は下側)の折り返し表示 0:無し、1:あり

Bit4-7 は常に 0 とし、BannarType = 0x10...ff は将来予約

BackColor :1byte

表示画像が無い部分の色。(ただし Image Chunk データ の表示オブジェクトにだけ有効)

WindowSpan :1～2byte Duration 表現を採用します。

表示領域の幅/高さ。

0 を指定した場合はバナー表示シーケンス(Auxiliary Sub-block)を無視します。

WindowSpan の値はレイアウト座標指定の座標決定に使用する ObjectWidth を変化させます。

Effective Span :1～2byte Duration 表現を採用します。

実際の画像の大きさとバナー表示の際の繰り返し周期を変更したい場合に追加します。

Effective Span は実際の画像のサイズより大きい値を指定しなければなりません。

元画像サイズ以下(0 を含む)を指定した場合は、元画像サイズと一致した値を指定したものとみ

なします。

水平バナーの場合は元画像の水平方向の実効幅、垂直バナーの場合は垂直方向の実効高を指定したことになります。

InitialPosition :1～2byte Coordval 表現を採用します。

初期表示位置。

BannerTravel :1～2byte Duration 表現を採用します。

バナー表示が終了するまえの移動距離。0なら指定無しと見なします。

BannerTime :1～2byte Duration 表現を採用します。

BannerTravel で指定した位置まで表示を動かすのに要する時間。この時間を超えると表示領域に表示された画像は停止したまま表示されます。BannerTime は TimeBase を基準として指定します。

BannerTravel = 0 の場合、BannerTime は 100dot 移動するのに要する時間を指定したものと解釈し、この速度のまま停止しないで表示消去時刻まで繰り返しバナー表示を続けます。

バナー表示基本概念

バナーは Primary Sub-block で表現した表示オブジェクトを「表示領域」に転送する方法と、 転送元となる画像上の位置の時間変化を指定することで定義します。距離や大きさは表示デバイスの 1 ドットを単位とします。

説明の便宜上、「バナー位置」という位置指定方法を定義します。バナー位置は画像を表示領域に転送する際の基準位置を定義する座標です。転送先の表示領域に対応する矩形を転送もとの画像上に配置した時の左上の座標をバナー位置と定義します。座標は画像の左上を原点としたローカル座標で表現します

バナー位置は画像の外側に対しても指定できます。

折り返しのある場合、元画像が隙間無くタイル状に並べられていると考えて画像を表示領域に転送します。つまり、バナー位置は画像サイズを単位として折り返して考えます。

折り返しの無い場合、画像が無い領域として塗り潰します。

表示オブジェクトが Text、Bitmap、Text Block の2値指定に対応する画像の場合はその表示オブジェクトの文字背景色で塗り潰します。BackColor は無視されます。

表示オブジェクトが Image、Image Tile の場合にはバナー表示 Sub-block 内の BackColor で塗り潰します。

バナータイプで元画像の折り返しかたに制限を加えることが可能です。

バナータイプで以下の機能を指定します。

□ バナー方向

水平バナー

画像の高さと同じ高さの表示領域に画像を投影し、水平方向に移動しながら表示します。
画面上において表示オブジェクトが右から左へ動くのを正方向、左から右へ動くのを負方向と定義します。

垂直バナー

画像の幅と同じ幅の表示領域に対して画像を投影し、垂直方向に移動しながら表示します。
画面上において表示オブジェクトが下から上へ動くのを正方向、上から下への動くのを負方向と定義します。

□　画像の折り返し方式の定義

負方向への折り返しの有無

バナー位置が負の値になったとき、画像を折り返して表示するかどうかを指定します。

正方向への折り返しの有無

バナー位置が画像の大きさより大きくなった時に画像を折り返して表示するかどうか指定します。
折り返しをしない場合、そこには元画像がないことになります。

折り返しの指定が無くバナーがその領域を表示する場合、画像が無い領域として以下のように塗り潰します。

表示オブジェクトが Text、Bitmap、Text Block の2値指定に対応する画像の場合は
その表示オブジェクトの文字背景色で塗り潰します。BackColor は無視されます。
表示オブジェクトが Image、Image Tile の場合にはバナー表示 Sub-block 内の
BackColor で塗り潰します。

□　画像の実効サイズの定義

画像を繰り返してバナー表示をする場合、間隔を開けて表示したい場合があります。そのような場合、画像の「実効サイズ」を指定して、実際より大きな画像として取り扱うことができます。

「実効サイズ」は元画像より大きくなくてはいけません。元画像をはみ出した部分を表示領域に転送する場合、元画像が存在しないので以下のように塗りつぶします。

表示オブジェクトが Text、Bitmap、Text Block の2値指定に対応する画像の場合は
その表示オブジェクトの文字背景色で塗り潰します。BackColor は無視されます。
表示オブジェクトが Image、Image Tile の場合にはバナー表示 Sub-block 内の
BackColor で塗り潰します。

テキストの場合にはスペース文字を挿入することで見かけ上同じ効果がえられます。
表示オブジェクトとしてのメモリー領域は大きくなります。

Effective Span は実際の画像のサイズより大きい値を指定しなければなりません。
ただし、元画像サイズ以下の値(0 を含む)を指定した場合は、元画像サイズと一致した値を指定し

たものとみなします。

□ バナー距離と所要時間

バナー距離はバナー開始時と終了時のバナー位置の差の絶対値です。

所要時間は指定したバナー距離を移動するのに要する時間です。

バナーが指定のバナー距離を移動した後停止します。

バナー距離の指定を 0 にするとバナーは止まりません。(表示消去時刻まで動き続けます)

バナー距離が 0 とした場合の所要時間は 100 ドット当たりの所要時間と見なします。

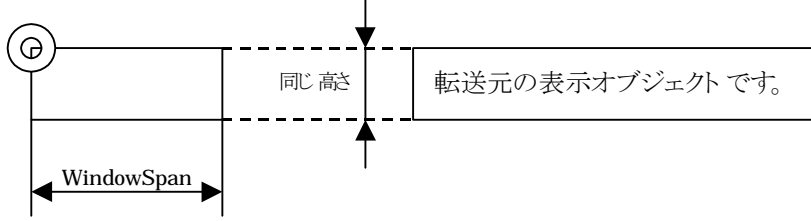
特別な値

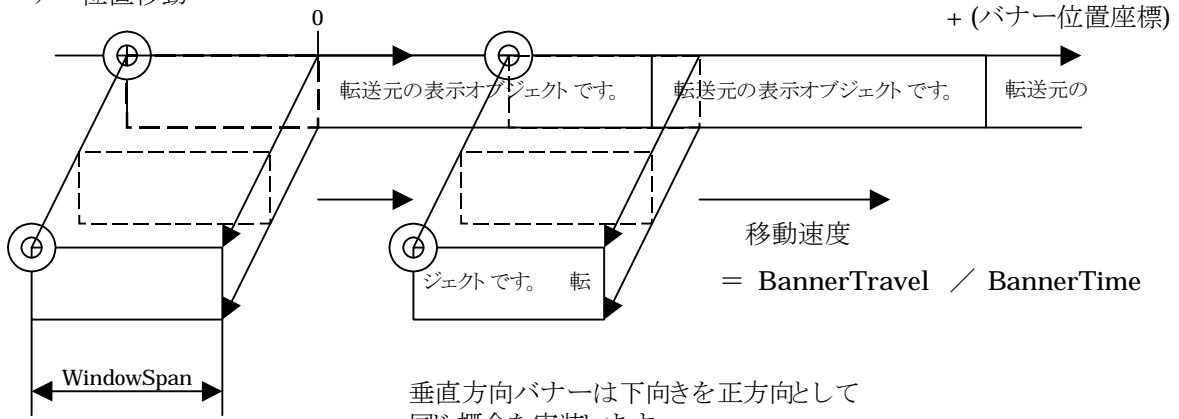
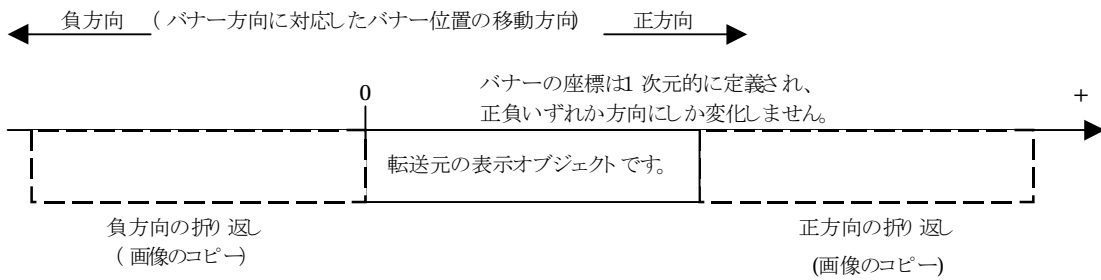
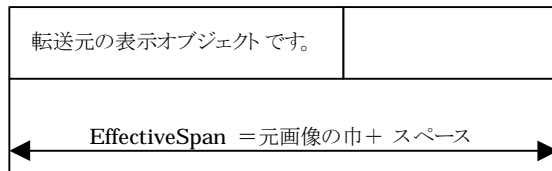
WindowSpan = 0x00 のときバナー表示シーケンスを無視します。

EffectiveSpan が元画像サイズより小さい場合元画像サイズ以下の値(0 を含む)を指定した場合は、元画像サイズと一致した値を指定したものとみなします。

BannerTime = 0x00 のときバナー表示シーケンスを無視します。

水平方向バナーの概念

画面上の表示領域 (左上丸印はバナー位置の貼り付け基準位置 (バナー方向によらない))

正方向に折り返し指定有りの正方向バナーの例
EffectiveSpan ＝0
InitialPosition ＝－WindowSpan

正方向バナーの場合バナー位置はローカル座標系を
左から右へ正方向へ移動します。
その結果オブジェクトは画面上 右から左へ移動します。

画像の折り返しは EffectiveSpan を含んだ大きさを折り返します。
EffectiveSpan は常にバナー座標の正方向に付加されます。(右側又は下側)

水平方向バナー概念図

- TravelSeq

表示オブジェクトの表示期間中に表示オブジェクトを移動させることができます。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x84							
Size	可変長							
Body	Step							
	Entry 数の 1byte 目							
	Entry 数の 2byte 目 [Option]							
	[以下 Entry 数だけ繰り返し]							
	1	1	1	1	xx		yy	
	目標 X 座標の 1byte 目							
	目標 X 座標の 2byte 目 [Option]							
	目標 Y 座標の 1byte 目							
	目標 Y 座標の 2byte 目 [Option]							
	到達時間の 1byte 目							
	到達時間の 2byte 目 [Option]							
	[繰り返し]							

Sub-block Type :=　0x84:TravelSeq

Sub-block Body :=

　Step　　　　　:1byte

　　表示更新間隔。0:=なるべくなめらかに、1...:=更新時間間隔

　Entry 数　　　:1～2byte　　Duration 表現を採用します。

　Entry 毎の構造{

　　目標座標　　:2～5byte　　Coordinates 表現を採用します。

　　到達時間　　:1～2byte　　Duration 表現を採用します。

　}「Entry 数」だけ繰り返し

到達時間は TimeBase を基準として指定します。

基本概念

Entry の中で指定された移動シーケンスを Step の情報に基づいて現在いるべき座標を線形補完して表示を更新します。

到達時間は TimeBase を単位として指定します。移動シーケンスは表示 Event の開始時刻を基準として相対時間で記述されます。到達時間は目標座標移動の時間間隔を指定します。到達時間は Duration 表現を採用します。

Entry 数と Data 数の対応

Entry=< Data 数　Entry 数の Data を解釈します。

Entry>　Data 数　Data を全て解釈し、Entry 数を減らします。

特殊な値

到達時間が 0x00 の場合、その Entry の時間に元の位置が消去され、新しい場所に移動したと解釈します。(瞬間的な移動)、ただし到達時間 0x00 の連続は 2 個目以降の Entry を無視します。目標座標が前後の Entry で同じ場合は到達時間だけその場所に停止しています。

- BlinkSeq

表示オブジェクトの表示期間中に表示オブジェクトを OnOff指定周期で点滅させることができます。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x85							
Size	0x03～0x06							
Body	OnTime の 1byte 目							
	OnTime の 2byte 目 [Option]							
	OffTime の 1byte 目							
	OffTime の 2byte 目 [Option]							
	Refrain の 1byte 目							
	Refrain の 2byte 目 [Option]							

Sub-block Type := 0x85:BlinkSeq

Sub-block Body:=

OnTime :1～2byte この時間を経過したら表示を消去します。表示 Event の開始時刻からの相対時間。 Duration 表現を採用します。

OffTime :1～2byte この時間を経過したら表示を再開します。Duration 表現を採用します。

Refrain :1～2byte 状態変化させる回数を指定します。0を指定すると表示オブジェクトの LifeTime の間繰り返します。Duration 表現を採用します。

OnTime OffTime とも TimeBase を基準として指定します。

基本概念

表示 Event の開始時刻から、表示オブジェクトが表示状態からシーケンスを開始し、OnTime を経過した時点で状態を非表示へ変更します。そのまま OffTime を経過した時点で再び状態を表示へ変更します。

Refrain は状態変化の回数を指定します。0を指定することで LifeTime 内の繰り返しを表現します。

特殊な値

OnTime および OffTime の少なくともどちらかが 0x00 の場合点滅表示シーケンスを無視します。

- ColorBlinkSeq

表示オブジェクトの表示期間中に OnOff指定周期で表示色を反転させることができます。

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x86							
Size	0x03～0x06							
Body	OnTime の 1byte 目							
	OnTime の 2byte 目 [Option]							
	OffTime の 1byte 目							
	OffTime の 2byte 目 [Option]							
	Refrain の 1byte 目							
	Refrain の 2byte 目 [Option]							

Sub-block Type := 0x86:ColorBlinkSeq

Sub-block Body :=

OnTime :1～2byte この時間を経過したら色換えを行います。表示 Event の開始時刻からの相対時間。 Duration 表現を採用します。

OffTime :1～2byte この時間を経過したら色を元に戻します。 Duration 表現を採用します。

Refrain :1～2byte 状態変化させる回数を指定します。0を指定すると表示オブジェクトの LifeTime の間繰り返します。 Duration 表現を採用します。

OnTime OffTime とも TimeBase を基準として指定します。

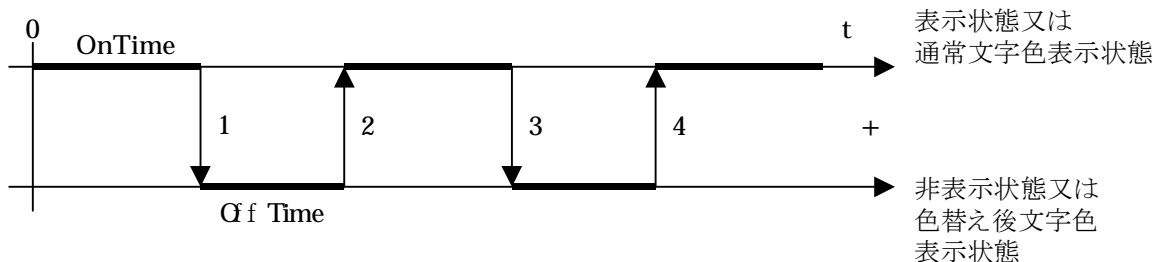
基本概念

表示 Event の開始時刻から、表示オブジェクトが文字色と文字背景色の表示状態からシーケンスを開始し、OnTime を経過した時点で状態を色替え後の文字色と色替え後の文字背景色へ色反転します。そのまま OffTime を経過した時点で再び状態を元の表示色へ変更します。

Refrain は状態変化の回数を指定します。0を指定することで LifeTime 内の繰り返しを表現します。

特殊な値

OnTime および OffTime の少なくともどちらかが 0x00 の場合カラー反転シーケンスを無視します。

点滅表示シーケンスおよび
カラー反転表示シーケンスにおける
OnTime OffTime Refrain の概念

Refrain = 0 なら LifeTime の期間繰り返しを行う。

点滅表示およびカラー反転シーケンス概念図

- Fill-InOutSeq

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x87							
Size	0x04～0x07							
Body	Fill-InType				Fill-OutType			
	Fill-InTime の 1byte 目							
	Fill-InTime の 2byte 目 [Option]							
	KeepTime の 1byte 目							
	KeepTime の 2byte 目 [Option]							
	Fill-OutTime の 1byte 目							
	Fill-OutTime の 2byte 目 [Option]							

Sub-block Type :=　0x87:Fill-InOutSeq

Sub-block Body :=

ShutterType	:1byte	オブジェクトの表示開始時のエフェクトと終了時のエフェクトを指定します。
Fill-InTime	:1～2byte	表示オブジェクトが全て表示されるまでの時間を指定します。
KeepTime	:1～2byte	表示オブジェクトが全て表示されている時間を指定します。
Fill-OutTime	:1～2byte	表示オブジェクトが全て表示されなくなる時間を指定します。

Fill-InTime、KeepTime、Fill-OutTime はいずれも Duration 表現を採用します。

Fill-InTime KeepTime、Fill-OutTime とも TimeBase を基準として指定します。

Fill-InType Fill-OutType	Description
0	エフェクト無し
1	Left-Horizontal In(Out)
2	Right-Horizontal In(Out)
3	Center-Horizontal In(Out)
4	Top-Vertical In(Out)
5	Bottom‐Vertical In(Out)
6	Center-Vertical In(Out)
7	Left-Top-Corner In(Out)
8	Left-Bottom-Corner In(Out)
9	Right-Top-Corner In(Out)
10	Right-Bottom-Corner In(Out)
11	Center-Rectangle In(Out)
12～15	Reserved

表示始めと表示終わりの位置を基準に動作名称を定義します。

基本概念

表示開始時と表示終了時に表示オブジェクトの表示領域を変化させるシーケンスを記述します。
ShutterType は上位 4bit により開始時の表示領域変化のタイプ、下位4bit により終了時の表示領域変化のタイプを指定します。

表示開始時は非表示状態から開始し Fill-InTime を経過した時刻で全体が表示されています。
表示終了時は Fill-InTime+KeepTime 時刻で全体が表示された状態から領域変化を開始し、Fill-OutTime を経過した時点で非表示の状態になります。

□ ShutterType

(In)

水平方向変化タイプ

Left-Horizontal-In　表示オブジェクトの左端から右端へ向かって表示を開始します。
Right-Horizontal-In　表示オブジェクトの右端から左端へ向かって表示を開始します。
Center-Horizontal-In　表示オブジェクトの中央から左右両端へ向かって表示を開始します。

垂直方向変化タイプ

Top-Vertical-In　表示オブジェクトの上端側から下端へ向かって表示を開始します。
Bottom-Vertical-In　表示オブジェクトの下端側から上端へ向かって表示を開始します。
Center-Vertical-In　表示オブジェクトの中央から上下端側へ向かって表示を開始します。

コーナー矩形変化タイプ

Left-Top-Corner-In　表示オブジェクトの左上から右下へ向かって表示を開始します。
Left-Bottom-Corner-In　表示オブジェクトの左下から右上へ向かって表示を開始します。
Right-Top-Corner-In　表示オブジェクトの右上から左下へ向かって表示を開始します。
Right-Bottom-Corner-In　表示オブジェクトの右下から左上に向かって表示を開始します。
Center-Rectangle-In　表示オブジェクトの中央から矩形状に表示を開始します。

(Out)

水平方向変化タイプ

Left-Horizontal-Out 表示オブジェクトの右端から左端へ向かって表示を終了します。
Right-Horizontal-Out 表示オブジェクトの左端から右端へ向かって表示を終了します。
Center-Horizontal-Out 表示オブジェクトの左右端から中央へ向かって表示を終了します。

垂直方向変化タイプ

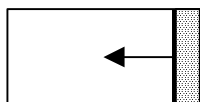
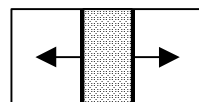
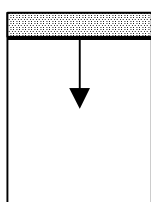
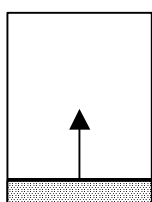
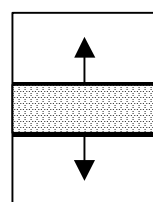
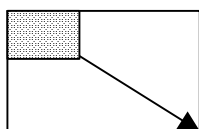
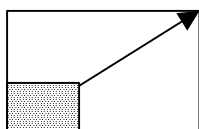
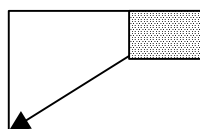
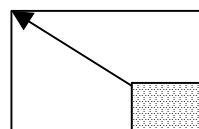
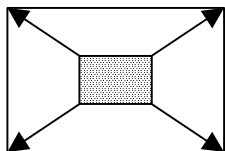
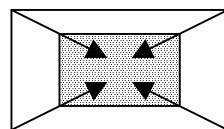
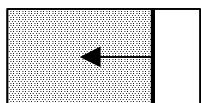
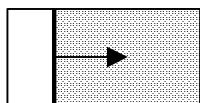
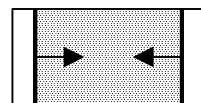
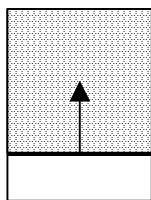
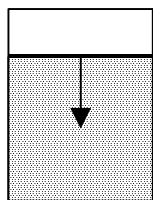
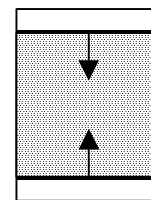
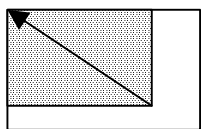
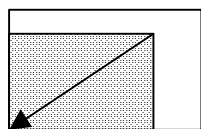
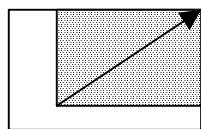
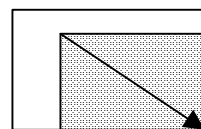
Top-Vertical-Out 表示オブジェクトの下端側から上端へ向かって表示を終了します。
Bottom-Vertical-Out 表示オブジェクトの上端側から下端へ向かって表示を終了します。
Center-Vertical-Out 表示オブジェクトの上下両端から中央へ向かって表示を終了します。

コーナー矩形変化タイプ

Left-Top-Corner-Out 表示オブジェクトの右下から左上へ向かって表示を終了します。
Left-Bottom-Corner-Out 表示オブジェクトの右上から左下へ向かって表示を終了します。
Right-Top-Corner-Out 表示オブジェクトの左下から右上へ向かって表示を終了します。
Right-Bottom-Corner-Out 表示オブジェクトの左上から右下に向かって表示を終了します。
Center-Rectangle-Out 表示オブジェクトの周辺から中央へ矩形状に表示を終了します。

特殊な値

ShutterType が 0x00(両方が指定無し)の場合 Fill-InOut シーケンスを無視します。
ShutterType が 0x0n 又は 0xn0 の場合は 0 側の指定が無いと解釈して Fill-InTime と Fill-OutTime の値によらず KeepTime に加算して開始側又は消滅側だけを処理します。
Fill-InTime 又は Fill-OutTime が 0x00 の場合は ShutterType の指定によらず Time 0 側の指定が無いと解釈します。

時間経過の概念
100%表示状態
t
非表示状態
Fill-InTime
KeepTime
Fill-OutTime
ShutterType の概念
Left-Horizontal-In
Right-Horizontal-In
Center-Horizontal-In
Top-Vertical-In
Bottom-Vertical-In
Center-Vertical-In
Left-Top-Corner-In
Left-Bottom-Corner-In
Right-Top-Corner-In
Right-Bottom-Corner-In
Center-Rectangle-In
ShutterType = 0
Fill-InTime = 0
Fill-OutTime = 0
の場合には Fill-InOut の指定無しと解釈する。
Center-Rectangle-Out
Left-Horizontal-Out
Right-Horizontal-Out
Center-Horizontal-Out
Top-Vertical-Out
Bottom-Vertical-Out
Center-Vertical-Out
Left-Top-Corner-Out
Left-Bottom-Corner-Out
Right-Top-Corner-Out
Right-Bottom-Corner-Out

3.) Display Parameter の適用範囲と解釈

- SMAF ファイルとしてのデフォルト設定

  Display Parameter Definition Chunk(§6.2.1)内の Event Type = 0x00 による指定(必須)を SMAF ファイルとしてのデフォルト設定とします。
  デフォルトの設定は同じ Graphics Track Chunk に対して有効です。

- Event Type 毎の個別設定

  デフォルト設定に対して、同じ Display Parameter Definition Chunk 内で定義することにより、Event Type 毎にオーバーライド(上書き)した設定が可能です。
  Sequence Data 内の Event Type にこの定義の存在する Event Type が指定された場合はその個別設定を選択します。
  Event Type 毎の個別設定は同じ Graphics Track Chunk に対して有効です。

- Parameter Override Sub-block による Event 設定

  Parameter Override Sub-block によって Sequence Data 内の Event の設定をオーバーライドした設定が可能です。(Event の内容解釈の前に有効とします。)
  Parameter Override は Event の頭から終わりの全て(Event、Primary Sub-block、Auxiliary Sub-block)に対してだけ有効です。

- 個別データ毎の指定(Coordinates)

  通常の Display Parameter はこれまでの設定による指定変更だけが可能です。
  Coordinates は、以上の3つの設定に対してさらに個別データのオーバーライドが可能です。
  Event の表示位置指定の Coordinates は Parameter Override の次に解釈され、変更があれば、その時点から Event の終了まで有効です。
  Auxiliary Sub-block 内のシーケンスデータ内の個別設定は、変更があればその時点から同じ Sub-block 内の次の設定までか、その Sub-block の最後までが有効です。
  (同じ Event の次の Sub-block は、Event の設定からスタートします。)

- Coordinatesの設定に対する運用ルール

  個別のオーバーライドが可能な設定を乱用することをさけるために、SMAF コンテンツ作成ガイドライン又は、SMAF 再生系実装ガイドラインでルールを設けて運用することとします。

4.)　Sub-block の対応一覧

ファイルフォーマットのデータ表現上の対応可否を一覧表にしました。

Auxiliary / Primary	Text	Bmp	Image	Rectangle	TextBlock	ImageTile	BitmapTile
Parameter Override	○	○	○(一部)	○(一部)	○	○(一部)	○(一部)
WipeTiming	○	○	×	×	○	×	×
WipeSeq	○	×	×	×	○	×	×
BannerInfo	○	○	○	○	○	○	○
TravelSeq	○	○	○	○	○	○	○
BlinkSeq	○	○	○	○	○	○	○
ColorBlinkSeq	○	○	×	×	○	×	×
Fill-InOutSeq	○	○	○	○	○	○	○

(一部については次の Display Parameter の対応一覧を参照してください。)

再生系およびコンテンツを含めて運用上の対応可否については別途、SMAF 再生系実装ガイドラインおよびコンテンツ製作のガイドラインにて設定します。

5.)　Display Paramter の対応一覧

ファイルフォーマットのデータ表現上の対応可否を一覧表にしました。

Display Parameter	Text	Bmp	Image	Rectangle	TextBlock	ImageTile	BitmapTile
Font Type	○	×	×	×	○	×	×
Font Size	○	×	×	×	○	×	×
Direction	○	×	×	×	○	×	×
Attribute	○	×	×	×	○	×	×
Font Color 0	○	○	×	×	○	×	○
Font Color 1	○	○	×	×	○	×	○
Edge Color 0	○	×	×	×	○	×	×
Edge Color 1	○	×	×	×	○	×	×
Back Color 0	○	○	×	×	○	×	○
Back Color 1	○	○	×	×	○	×	○
Coordinates	○	○	○	○	○	○	○
BackDrop Color	×	×	×	×	×	×	×
Transparent Color	○	○	○	○	○	○	○
Transparent Enable	○	○	○	○	○	○	○

再生系およびコンテンツを含めて運用上の対応可否については別途、SMAF 再生系実装ガイドラインおよびコンテンツ製作のガイドラインにて設定します。

6.4. Font Data Chunk

Chunk ID	"Gftd"	:Font Data Chunk

Font Data Chunk の内容は外字 Bitmap のデータを格納した Font Chunk の配列です。

6.4.1. Font Chunk

Chunk ID	"Ge**”	:Font Chunk

外字データのリソース Chunk です。

Font Chunk は FontType と FontSize で一つの Chunk にまとめられ、Code によって検索して使用します。

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	0x47 (“G”)							
Data#1	0x65 (“e”)							
Data#2	FontType							
Data#3	FontSize							

Font Chunk は

Header

FontData　　外字データの配列(可変長データ配列)

から構成されます。

Header は

EntryNumber :1～2byte　　Chunk 内の外字数 Duration 表現を採用します。

から構成されます。

外字一文字毎のデータ構造は

```
FontData {
    Code      :2byte
    Size      :1～2byte   Duration 表現を採用します。
    Bitmap    :可変長
}
```

FontData を外字の数繰り返します。

FontType および FontSize は表示オブジェクトの Display Parameter の指定された値によって指定の Chunk を検索します。Chunk 内の該当のコードを使用して表示します。

フォーマット上は Chunk 内の外字 Bitmap の幅方向ドットサイズを規定しません。
再生系は得られた Bitmap をドットのスペースを挟まず、連続して並べて表示します。

実際の携帯端末上での当初の外字運用ルールは実装ガイドラインに指定します。
使用する Bitmap のデータフォーマットの定義は Bmp Chunk に記述します。

6.4.2. Unicode Font Chunk

Chunk ID　"Gu**"　:Unicode Font Chunk

外字データのリソース Chunk です。
Font Chunk は FontType と FontSize で一つの Chunk にまとめられ、Code によって検索して使用します。

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	0x47 (“G”)							
Data#1	0x75 (“u”)							
Data#2	FontType							
Data#3	FontSize							

Font Chunk は
　Header
　FontData　　外字データの配列(可変長データ配列)
から構成されます。

Header は
　EntryNumber :1～2byte　　Chunk 内の外字数 Duration 表現を採用します。
から構成されます。

外字一文字毎のデータ構造は
FontData {
　Size　　: 1～2byte　Duration 表現を採用します。
　Code　　:可変長
　Size　　:1～2byte　Duration 表現を採用します。
　Bitmap　:可変長
}

FontData を外字の数繰り返します。
FontType および FontSize は表示オブジェクトの Display Parameter の指定された値によって指定の Chunk を検索します。Chunk 内の該当のコードを使用して表示します。

それぞれの Code 先頭に BOM (バイトオーダーマーク) を設定すること。
BOM 無しの場合、ビッグエンディアンとして解釈すること。

フォーマット上は Chunk 内の外字 Bitmap の幅方向ドットサイズを規定しません。
再生系は得られた Bitmap をドットのスペースを挟まず、連続して並べて表示します。

実際の携帯端末上での当初の外字運用ルールは実装ガイドラインに指定します。
使用する Bitmap のデータフォーマットの定義は Bmp Chunk に記述します。

6.5. Image Data Chunk

Chunk ID "Gimd" :Image Data Chunk

Image Data Chunk の内容は画像情報 Chunk の配列です。
画像情報 Chunk には以下の 3 種類があります。

Image Chunk
Bmp Chunk
Link Chunk

画像情報 Chunk の ChunkID の下位1byte は Image Data Chunk 内で
ユニークな番号を保持します。この ID が Display Object Event で指定する「画像 ID」
となります。

6.5.1. Image Chunk

Chunk ID "Gig*" :Image Chunk *=0x00 ～ 0xFF

JPEG、PNG 形式などの汎用画像フォーマットのデータを格納します。
画像データは汎用表示イベントに Primary の Sub-block Type に Image を指定して使用します。

6.5.2. Bmp Chunk

Chunk ID	"Gbm*"	:Bmp Chunk	*=0x00 ～ 0xFF

Bitmap（2 値画像）のデータを格納します。

Bitmap データは汎用表示イベントに Primary の Sub-block Type に Bitmap を指定して使用します。

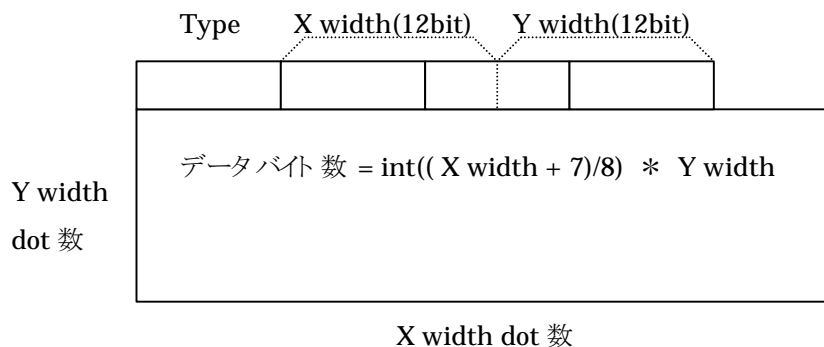
Bitmap のデータフォーマット

Type (1byte)

Size (3byte = X 方向 dot 数 12bit 表現 , Y 方向 行数 12bit 表現)

Data (1dot あたり 1bit による定義、サイズによる可変長)

Type	Description
0x00	無圧縮
0x01	8bit Run Length 圧縮
0x01～FF	Reserved

Size	b7	b6	b5	b4	b3	b2	b1	b0
Data #1	Xwidth（上位 8bit）							
Data #2	Xwidth (下位 4bit)				Ywidth (上位 4bit)			
Data #3	Ywidth (下位 8bit)							

データの MSB が画面の左上角を指定します。

ファイル終端に存在する余りの bit には 0 を入れておきます。

SMAF で使用する PC 上の2値画像ファイルの拡張子を ＊.bms (BitMap of Smaf) とします。

Run Length の圧縮アルゴリズム (8bit Run Length)

データをバイト単位にみて,
同じデータが n 回続いたとき (n は 2 〜 129),
byte ^(n - 2)
byte val
・違うデータが n 回続いたとき (n は 1 〜 128),
byte n - 1
byte val1, val2, val3..., valn

Ex.) 00 00 00 00 00 05 05 01 02 ==> fc 00 ff 05 01 01 02

6.5.3. Link Chunk

Chunk ID "Gln*" :Link Chunk *=0x00 〜 0xFF

SMAF ファイルの外部にある画像を指定するのに使用します。
詳細定義は未定です。

7. Master Track 基本概念

7.1. Master Track の位置付け

Master Track は SMAF 仕様データの中で、音楽の情報を記述するシーケンスと SMAF 再生系そのものを制御するためのシーケンスを表現するデータ表現です。
Master Track は SMAF を構成する Track の一つですが、Score Track PCM Track、Graphics Track と異なり、対応する出力デバイスを持っていません。

7.2. Master Track の基本構造

Master Track は

- Header
- Sequence Data Chunk

から構成されています。
Sequence Data Chunk は、他の Track の Sequence Data と同様に Duration、Event、EOS から構成されます。

Event が表現する内容として、以下のように大きく二つに分類することができます。
Event の種類

- 音楽情報 Event
- 演奏制御 Event

7.2.1. 音楽情報 Event

音楽情報 Event は和音名、調性、拍子、テンポ、小節線、リハーサルマークに対応する Event です。
再生される音楽のシーケンスに同期して情報 Event を貼りつけることにより音楽の構造を表現したり、
その時間における音楽情報を取り出すことができます。

7.2.2. 演奏制御 Event

演奏制御 Event は SMAF 再生系自身をコントロールする Event です。
例えば、Pause や Jump(分岐指定)などを想定しています。

(実装例)
演奏制御 Event に対応するために、再生系は Seek 機能やキー入力処理などの GUI 処理を
実装する必要があります。

例えば再生系が Master Track の Pause Event を処理した場合には、その時間で再生を一時停止します。
GUI からのいずれかのキー入力により再生を開始します。入力により時間を Seek して処理することも

可能です。

本仕様解説書においては詳細は未定義です。将来拡張とします。

本仕様解説書の内容を実装した SMAF(表示)再生系は Master Track Chunk の Sequnece Data を解釈しません。

7.2.3. 音楽情報 Event のガイドライン

1曲あたりの Master Track の容量は Score Track の 10〜20%以内に収まるようにすること。

可能な範囲で XF フォーマットのデータ表現から単純な変換が可能であること。

XF->MasterTRACK 変換が容易であること。逆変換は必ずしも可能でなくてもよい。

表現すべき内容と運用ルール

TimeSignature

TimeSingature は曲の先頭で指定しなければなりません。

TimeSignature は小節線タイミングでしか変更してはなりません。

小節線タイミング以外で指定されたら、次の小節線タイミングで指定されたものとみなします。

Key Signature

KeySignature を指定すべきタイミングの規定は TimeSignature に準じます。

Tempo

最低テンポを 40 と仮定すると、21bit で表現可能。7bit×3 で表現可能。

SMF のデータ表現が 24bit 分あり、変換失敗を避けるため 7bit×4=28bit 表現も定義します。

テンポ変更を指定できるタイミングの規定は TimeSignature に準じます。

Measure Mark

データ量を少なくするために小節線の位置だけを記録します。

Beat タイミングは TimeSignature と小節線の位置から推測することになります。

Chord Name

XF のコード名指定を基本に、通常は分数コード指定を省略してデータ量を減らせる形式にします。

運用上指定可能なコード名は別途定めます。

8. Master Track データフォーマット

8.1. Master Track Chunk

Master Track はシーケンサが曲の構造や情報を認識したり、時間の流れを制御するのための情報を格納する Track です。拡張予定の機能は以下の通りです。

Chunk ID	"MSTR"	:Master Track Chunk

Master Track の内部構造は以下の通りです。

- Header
- Sequence Data Chunk

Header の内容

- FormatType　:1byte
- Sequence Type　:1byte
- TimeBase_D　:1byte
- OptionSize　:1byte
- OptionData　:OptionSize で指定したサイズ(0～255byte)

1. Format Type

使用するマスタートラックデータのデータフォーマットを定義します。

Format Type	Description
0x00	Handy Phone Standard
0x01～0xFF	Reserved

2. Sequence Type

Sequence Type	Description
0x00	Stream Sequence
0x01	Sub-Sequence
0x02～0xFF	Reserved

Stream Sequence

Score Track の Sequence Data は1つの連続したシーケンスデータで表現されています。

Sub-Sequence

Score Track の Sequence Data は複数のフレーズデータを連続で表記したものです。

3. TimeBase_D

TimeBase_D	Description
0x00	1 msec
0x01	2 msec
0x02	4 msec
0x03	5 msec
0x04～0x0F	Reserved
0x10	10 msec
0x11	20 msec
0x12	40 msec
0x13	50 msec
0x14～0xFF	Reserved

Score Track の Duration 基準時間の TimeBase_D に準じます。

4. OptionSize

直後の拡張用 OptionData のサイズを指定します。通常 OptionSize:= 0x00 とします。
拡張時のサイズは 4 の倍数とします。

5. OptionData

将来拡張用のデータを記述します。解釈時に 0 を仮定してはいけません。

8.2. Master Track Sequence Data Chunk

Chunk ID "Mssq" :Master Track Sequence Data Chunk

Sequence Data 表現

Sequence Data は可変長バイトストリームです。
Sequence Data の構成要素は Duration と Event と EOS です。
Sequence Data の先頭要素は Duration とし、Duration と Event は交互に表記します。
EOS は Sequence の終了(End Of Sequence)の意味。4byte 以上の 0x00 の連続で表現します。

Sequence Data の解釈にあたっては Event、Duration の解釈に先だって EOS の確認をします。
0x00 が4byte 以上連続する Event や Duration が存在しないことをデータフォーマットとして
保証します。

Duration の表現

Duration の表現は Score Track、Graphics Track に準じます。

8.3. Event

Event のデータ表現

Event は 1 バイト以上の可変長とします。

可変長要素の最終バイトは MSB=0,それ以外のバイトは MSB=1 とします。

Event は常に MSB=0 のデータの出現を基準に終端を判断します。

既知の Event でも将来拡張が行われ、データ長が増える可能性があります。(データ長が減少することはありません。)

未知の Event は MSB=0 のデータまで(MSB=0 のデータを含む)読み飛ばします。

データ長の上限は規定していません。

No Operation Event (NOP Event)

何もしません。Score Track、Graphics Track に準じて、Duration の分割に使用します。

NOP Event = 0x00

8.3.1. 音楽情報 Event

最初に規定する音楽情報 Event の種類は以下の通りです。

1. Chord Name (コード名)

コード名称をルートノート(C...B)、変位記号(#,b),コードタイプを2バイトで表現します。

XF 表現に準拠します。

Data Count	b7	B6	b5	b4	b3	b2	b1	b0
Data #1	1	0	変位記号			ルート音名		
Data #2	S	コードタイプ(XF 準拠)						
[Data #3]	1	0	変位記号			ルート音名		
[Data #4]	0	コードタイプ(XF 準拠)						

分数コードを表現したい場合はコード名称を2つ連続して表記し、4バイトで表現します。

S	Description
0	2byte 表現
1	分数コードの 4byte 表現

データ構造

1byte 目および 3byte 目 (3bit による変位記号とルート音名)

変位記号(3bit)	Description
0	♭♭♭
1	♭♭
2	♭
3	ナチュラル
4	♯
5	♯♯
6	♯♯♯
7	使用禁止

ルート音名(3bit)	Description
0	使用しない(XF Reserved)
1	C
2	D
3	E
4	F
5	G
6	A
7	B

2byte 目および4byte 目(7bit による Chord Type)

Value	Type	Value	Type	Value	Type	Value	Type
0	Maj	9	min6	18	dim7	27	7(#9)
1	Maj6	10	min7	19	7th	28	Maj7aug
2	Maj7	11	Min7b5	20	7sus4	29	7aug
3	Maj7(#11)	12	Min(9)	21	7b5	30	1+8
4	Maj(9)	13	Min7(9)	22	7(9)	31	1+5
5	Maj7(9)	14	Min7(11)	23	7(#11)	32	sus4
6	Maj6(9)	15	MinMaj7	24	7(13)	33	1+2+5
7	aug	16	MinMaj7(9)	25	7(b9)	34	cc
8	min	17	dim	26	7(b13)	35～127	指定なし

2.　KeySignature (調性情報)

KeySignature を 2 バイトで表現します。

データ構造

Data Count	b7	B6	b5	b4	b3	b2	b1	b0
Data #1	1	0	1	1	1	0	0	0
Data #2	0	0	0	m	変化記号の数			

変化記号の数 := #,b の数。4bit 符号付き2進数として解釈。>0 は#の数、<0 は b の数。

m	Description
0	長調
1	短調

3. Time Signature (拍子情報)

Time Signature を2バイトで表現します。

データ構造

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #1	1	0	1	1	1	fff 分母		
Data #2	0	nnnnnnn 分子						

1 バイト目:= 0xb9～0xbd

Data fff	分母
0	使用禁止
1	2
2	4
3	8
4	16
5	32
6	未定義
7	未定義

未定義値が指定されたら TimeSignature イベントを無視します。

0 は KeySignature と同じコードになってしまうため禁止します。

2 バイト目:= 0nnnnnnn

nnnnnnn: 拍子の分子。1...64 が有効範囲とします。

4. Tempo (テンポ)

1拍の長さをマイクロ秒単位で表現。データ長は 4 または 5 バイト。

データ構造

Data Count	b7	B6	b5	b4	b3	b2	b1	b0
Data #1	1	1	1	1	0	0	0	0
Data #2	1	上位 7bit						
Data #3	1	中位 7bit						
Data #4	S	下位 7bit						
[Data #5]	0	Option 最下位 7bit						

S	Description
0	4byte 表現
1	5byte 表現

4バイト表現(21bit 未満で 1 拍の長さが表現できる場合に使用)
5 バイト表現(21bit 未満で 1 拍の長さが表現できない場合に使用)

4バイト表現で記述可能なテンポ値を5バイト表現で表記することを禁止はしませんが推奨できません。

5. Measure Mark (小節線)

小節の区切れ位置を表します。1バイトで表現します。

データ構造

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #1	0	1	1	1	0	0	0	1

1バイト目:= 0x71

6. Rehearsal Mark(リハーサルマーク)

XF フォーマットに対応したリハーサルマークを表現します。

データ構造

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #1	0	1	0	0	xxxx			

1バイト目:= 0100xxxx

xxxx: 0:Intro,1:Ending,2:Fill-in,3:A,....,15:M

Data xxxx	リハーサルマーク
0	Intro
1	Ending
2	Fill-in
3	A
4	B
….	….
….	….
15	M

8.3.2. 演奏制御 Event

演奏制御 Event は詳細未定義です。

最初に定義すべき Event は Pause Event です。

9. Appendix

9.1. Chunk ID & TAG 一覧

SMAF 基本機能 Chunk ID & TAG 名一覧表

Track	Sub	S-Sub	TAG	Description
MMMD				Mobile Application Data Chunk
CNTI				Contents Information Chunk
			VN	Vender Name
			CN	Carrier Name
			CA	Category Name
			ST	Song Title
			AN	Artist Name
			WW	Writer's Name
			SW	Song writer's Name
			AW	Arranger writer's Name
			CR	Copyright
			GR	Group
			MI	Management Infomation
			CD	Created date
			UD	Updated date
OPDA				Optional Data Chunk
	Dch*			Data Chunk
			VN	Vender Name
			CN	Carrier Name
			CA	Category Name
			ST	Song Title
			AN	Artist Name
			WW	Writer's Name
			SW	Song writer's Name
			AW	Arranger writer's Name
			CR	Copyright
			GR	Group
			MI	Management Infomation
			CD	Created date
			UD	Updated date
			ES	CopyStatus after edit
			VC	virtual Card
MTR*				Score Track Chunk
	MspI			Score Seek & Phrase Info
			st	Start
			sp	Stop
			Pa	Phrase "a"
				:
			Pz	Phrase "z"
			PA	A-melo
			PB	B-melo
			PE	Ending
			PI	Intro
			PK	KANSOU
			PS	SABI
			PR	Refrain
			SL	Sub-sequence List
	Mtsu			Score Track Setup Data Chunk
	Mtsq			Score Track Sequence Data Chunk

	Mtsp			Score Track Stream PCM Data Chunk
		Mwa*		Score Track Stream Wave Data Chunk
ATR*				Audio Track Chunk
	AspI			Audio Track Seek & Phrase Info
	Atsu			Audio Track Setup Data Chunk
	Atsq			Audio Track Sequence Data Chunk
	Awa*			Audio Track Wave Data Chunk

(*) 大文字と小文字を区別する。

SMAF 拡張機能　Chunk ID & TAG 名一覧表 つづき

Track	Sub	S-Sub	Description
GTR*			Graphics Track Chunk
	Gtsu		Setup Data Chunk
		Gdpd	Display Parameter Definition Chunk
		Gcpd	Color Palette Definiton Chunk
	Gsq*		Graphics Track Sequence Data Chunk
	Gftd		Font Data Chunk
		Ge**	Font Chunk　(FontType　FontSize の区別)
		Gu**	Unicode Font Chunk (FontType Size の区別)
	Gimd		Image Data Chunk
		Gig*	Image Chunk
		Gbm*	Bmp Chunk
		Gln*	Link Chunk
MSTR			Master Track Chunk
	Mssq		Master Track Sequence Data Chunk

(*) 大文字と小文字を区別する。

9.2. CRC サンプルコード

CRC のサンプルコードを C で記述する。
n byte の byte 列 c の CRC 値を求めるサンプルである。
このサンプルでは予めテーブルを作成し高速化している。

```
#define CHAR_BIT     8       /* number of bits in a char */
#define UCHAR_MAX    0xff     /* maximum unsigned char value */
typedef unsigned charUint8;
typedef unsigned short  Uint16;
static Uint16 crctable[UCHAR_MAX + 1];
#define CRCPOLY1  0x1021U

void  makeCRCTable(void)
{
   Uint16 i, j, r;
   for (i=0; I <= UCHAR_MAX; i++) {
      r = i << (16-CHAR_BIT);
      for (j=0; j < CHAR_BIT; j++) {
      if (r & 0x8000U) {
            r = (r<<1) ^ CRCPOLY1;
         } else {
            r <<= 1;
         }
      }
      crctable[i] = r&0xFFFFU;
   }
}

Uint16 makeCRC(int n, Uint8* c)
{
   Uint16  r;
   r = 0xFFFFU;
   while (--n >= 0) {
      r = (r << CHAR_BIT) ^ crctable[(Uint8)(r >> (16 - CHAR_BIT)) ^ *c++];
   }
   return ~r & 0xFFFFU;
}
```

9.3. BNF 表記

SMAF の BNF 表記を以下に記述する。

```
;--------------------------------------------------------------------------------
;          SMAF     (BNF  表記)
;
;   * : 0  回以上, + : 1  回以上, ? : 0  回  or 1  回
;--------------------------------------------------------------------------------

BYTE = [¥x00-¥xff]                          ; byte data  の定義
CHAR = [¥x00-¥x2b¥x2d-¥xff]        ; char data  の定義
ChunkSize = (BYTE BYTE BYTE BYTE)

;----- File Chunk (4.1項) ---------------------------------------------------------

Smaf_Chunk = "MMMD" ChunkSize   Contents_Info_Chunk Optional_Data_Chunk?
                (Score_Track_Chunk? | PCM_Audio_Track_Chunk? |
                Graphics_Track_Chunk?)+

;----- Contents Info Chunk (4.2項) -----------------------------------------------

Contents_Info_Chunk = "CNTI" ChunkSize
                              Contents_Class Contents_Type
                              Contents_Code_Type Copy_Status Copy_Counts
                             ((ContentsTag ":" CHAR) ("," ContentsTag ":" CHAR )*)?
   Contents_Class         = BYTE
   Contents_Type              = BYTE
   Contents_Code_Type      = BYTE
   Copy_Status             = BYTE
   Copy_Counts             = BYTE
   CotentsTag = ("VN"|"CN"|"CA"|"ST"|"AN"|"WW"|"SW"|"AW"|"GR"|"MI"|"CR"|"CD" |"UD")

;----- Optional Data Chunk (4.3項) -----------------------------------------------

Optional_Data_Chunk = "OPDA" ChunkSize Data_Chunk*

   Data_Chunk = "Dch" [¥x00-¥xff] ChunkSize (Contents_Tag Data_Size Data)*
   CotentsTag = BYTE BYTE
   Data_Size = BYTE BYTE
   Data = BYTE*

;----- Score Track Chunk (4.4項) -------------------------------------------------

Score_Track_Chunk = "MTR" [¥x00-¥xff] ChunkSize
                 Format_Type Sequence_Type Timebase_D
                 Timebase_G Channel_Status
                (Seek&Phrase_Info_Chunk? Setup_Data_Chunk? Sequence_Data_Chunk |
                 Seek&Phrase_Info_Chunk? Sequence_Data_Chunk Setup_Data_Chunk? |
                 Setup_Data_Chunk? Seek&Phrase_Info_Chunk? Sequence_Data_Chunk |
                 Setup_Data_Chunk? Sequence_Data_Chunk Seek&Phrase_Info_Chunk? |
                 Sequence_Data_Chunk Setup_Data_Chunk? Seek&Phrase_Info_Chunk? |
                 Sequence_Data_Chunk Seek&Phrase_Info_Chunk? Setup_Data_Chunk? )
   Format_Type  = BYTE

   ;      for Handy Phone Standard Format (Format_Type=0x00)

   Sequence_Type     = BYTE
   Timebase_D         = BYTE
```

```
    Timebase_G        = BYTE
    Channel_Status = BYTE BYTE
    Seek&Phrase_Info_Chunk = "MsqI" ChunkSize
          ( (StartPoint4 "," StopPoint4 (PhraseList)?)
           |((StartPoint1 "," StopPoint1 PhraseList SubsequenceList)? );
    StartPoint1 = "st:" BYTE;
    StopPoint1   = "sp:" BYTE;
    StartPoint4 = "st:" BYTE BYTE BYTE BYTE;
    StopPoint4   = "sp:" BYTE BYTE BYTE BYTE;
    PhraseList = (","
                    ("Pa"|"Pb"|"Pc"|"Pd"|"Pe"|"Pf"|"Pg"|
                     "Ph"|"Pi"|"Pj"|"Pk"|"Pl"|"Pm"|"Pn"|
                     "Po"|"Pp"|"Pq"|"Pr"|"Ps"|"Pt"|"Pu"|
                     "Pv"|"Pw"|"Px"|"Py"|"Pz"|
                     "PA"|"PB"|"PE"|"PI"|"PK"|"PS"|"PR")
                    ":" BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE))*;
    SubsequenceList = ("," "SL:" [a-zABEIKSR]+ );
    Setup_Data_Chunk = "Mtsu" ChunkSize BYTE+
    Sequence_Data_Chunk = "Mtsq" ChunkSize
                         ((Duration Event)|(¥x00 ¥x00 ¥x00 ¥x00))+
    Duration = BYTE BYTE?
    Event       = BYTE BYTE*

;      for   Mobile Standard Format (Format_Type=0x01,0x02)

    Sequence_Type    = BYTE
    Timebase_D        = BYTE
    Timebase_G        = BYTE
    Channel_Status = BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE
                         BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE
    Seek&Phrase_Info_Chunk = "MsqI" ChunkSize
          ( (StartPoint4 "," StopPoint4 (PhraseList)?)
           |((StartPoint1 "," StopPoint1 PhraseList SubsequenceList)? );
    StartPoint1 = "st:" BYTE;
    StopPoint1   = "sp:" BYTE;
    StartPoint4 = "st:" BYTE BYTE BYTE BYTE;
    StopPoint4   = "sp:" BYTE BYTE BYTE BYTE;
    PhraseList = (","
                    ("Pa"|"Pb"|"Pc"|"Pd"|"Pe"|"Pf"|"Pg"|
                     "Ph"|"Pi"|"Pj"|"Pk"|"Pl"|"Pm"|"Pn"|
                     "Po"|"Pp"|"Pq"|"Pr"|"Ps"|"Pt"|"Pu"|
                     "Pv"|"Pw"|"Px"|"Py"|"Pz"|
                     "PA"|"PB"|"PE"|"PI"|"PK"|"PS"|"PR")
                    ":" BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE))*;
    SubsequenceList = ("," "SL:" [a-zABEIKSR]+ );
    Setup_Data_Chunk = "Mtsu" ChunkSize BYTE+
    Sequence_Data_Chunk = "Mtsq" ChunkSize
                         ((Duration Event)|(¥x00 ¥x00 ¥x00 ¥x00))+
    Duration = BYTE BYTE?
    Event       = BYTE BYTE*
    Stream_PCM_Data_Chunk = "Mtsp" ChunkSize Stream_Wave_Data_Chunk+
      Stream_Wave_Data_Chunk = "Mwa" [¥x00-¥xff] ChunkSize BYTE+

;----- PCM Audio Track Chunk(4.5項) ---------------------------------------------

PCM_Audio_Track_Chunk = "ATR" [¥x00-¥xff] ChunkSize
                 Format_Type Sequence_Type Wave_Type
                 Timebase_D Timebase_G
       (Seek&Phrase_Info_Chunk? Setup_Data_Chunk? Sequence_Data_Chunk Wave_Data_Chunk+|
```

```
        Seek&Phrase_Info_Chunk? Setup_Data_Chunk? Wave_Data_Chunk+ Sequence_Data_Chunk |
        Seek&Phrase_Info_Chunk? Sequence_Data_Chunk Setup_Data_Chunk? Wave_Data_Chunk+ |
        Seek&Phrase_Info_Chunk? Sequence_Data_Chunk Wave_Data_Chunk+ Setup_Data_Chunk? |
        Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ Sequence_Data_Chunk Setup_Data_Chunk? |
        Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ Setup_Data_Chunk? Sequence_Data_Chunk |
        Setup_Data_Chunk? Seek&Phrase_Info_Chunk? Sequence_Data_Chunk Wave_Data_Chunk+ |
        Setup_Data_Chunk? Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ Sequence_Data_Chunk |
        Setup_Data_Chunk? Sequence_Data_Chunk Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ |
        Setup_Data_Chunk? Sequence_Data_Chunk Wave_Data_Chunk+ Seek&Phrase_Info_Chunk? |
        Setup_Data_Chunk? Wave_Data_Chunk+ Seek&Phrase_Info_Chunk? Sequence_Data_Chunk |
        Setup_Data_Chunk? Wave_Data_Chunk+ Sequence_Data_Chunk Seek&Phrase_Info_Chunk? |
        Sequence_Data_Chunk Setup_Data_Chunk? Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ |
        Sequence_Data_Chunk Setup_Data_Chunk? Wave_Data_Chunk+ Seek&Phrase_Info_Chunk? |
        Sequence_Data_Chunk Seek&Phrase_Info_Chunk? Setup_Data_Chunk? Wave_Data_Chunk+ |
        Sequence_Data_Chunk Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ Setup_Data_Chunk? |
        Sequence_Data_Chunk Wave_Data_Chunk+ Setup_Data_Chunk? Seek&Phrase_Info_Chunk? |
        Sequence_Data_Chunk Wave_Data_Chunk+ Seek&Phrase_Info_Chunk? Setup_Data_Chunk?)
    Format_Type  = BYTE
    Sequence_Type    = BYTE
    Wave_Type        = BYTE
    Timebase_D       = BYTE
    Timebase_G       = BYTE

;     for Handy Phone Standard Format (4.5.1項)

    Seek&Phrase_Info_Chunk = "AsqI" ChunkSize
         ( (StartPoint4 "," StopPoint4 (PhraseList)?)
          |((StartPoint1 "," StopPoint1 PhraseList SubsequenceList)? );
    StartPoint1    = "st:" BYTE;
    StopPoint1     = "sp:" BYTE;
    StartPoint4    = "st:" BYTE BYTE BYTE BYTE;
    StopPoint4     = "sp:" BYTE BYTE BYTE BYTE;
    PhraseList         = (","
                  ("Pa"|"Pb"|"Pc"|"Pd"|"Pe"|"Pf"|"Pg"|
                   "Ph"|"Pi"|"Pj"|"Pk"|"Pl"|"Pm"|"Pn"|
                   "Po"|"Pp"|"Pq"|"Pr"|"Ps"|"Pt"|"Pu"|
                   "Pv"|"Pw"|"Px"|"Py"|"Pz"|
                   "PA"|"PB"|"PE"|"PI"|"PK"|"PS"|"PR")
                  ":" BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE))*;
    SubsequenceList  = ("," "SL:" [a-zABEIKSR]+ );
    Setup_Data_Chunk = "Atsu" ChunkSize BYTE+
    Sequence_Data_Chunk = "Atsq" ChunkSize
                      ((Duration Event) | (¥x00 ¥x00 ¥x00 ¥x00))+
    Duration = BYTE BYTE?
    Event       = BYTE BYTE*
    Wave_Data_Chunk = "Awa" [¥x00-¥xff] ChunkSize BYTE+

= BYTE

;----- Graphics Track Chunk (6.1項) -----------------------------------------------

Graphics_Track_Chunk = "GTR" [¥x00-¥xff] ChunkSize
         Format_Type Player_Type Text_Encode_Type Color_Type
         Timebase Option_Size Option_Data
        (Setup_Data_Chunk Sequence_Data_Chunk+  Font_Data_Chunk?  Image_Data_Chunk? |
          Setup_Data_Chunk Sequence_Data_Chunk+  Image_Data_Chunk?  Font_Data_Chunk?  |
          Setup_Data_Chunk Image_Data_Chunk?  Sequence_Data_Chunk+  Font_Data_Chunk?  |
          Setup_Data_Chunk Image_Data_Chunk?  Font_Data_Chunk?  Sequence_Data_Chunk+  |
          Setup_Data_Chunk Font_Data_Chunk?  Sequence_Data_Chunk+  Image_Data_Chunk?  |
```

```
        Setup_Data_Chunk Font_Data_Chunk?  Image_Data_Chunk?  Sequence_Data_Chunk+  |
        Sequence_Data_Chunk+  Setup_Data_Chunk Font_Data_Chunk?  Image_Data_Chunk?  |
        Sequence_Data_Chunk+ Setup_Data_Chunk  Image_Data_Chunk?  Font_Data_Chunk?  |
        Sequence_Data_Chunk+  Image_Data_Chunk?  Setup_Data_Chunk Font_Data_Chunk?  |
        Sequence_Data_Chunk+  Image_Data_Chunk?  Font_Data_Chunk?  Setup_Data_Chunk |
        Sequence_Data_Chunk+  Font_Data_Chunk?  Setup_Data_Chunk  Image_Data_Chunk? |
        Sequence_Data_Chunk+  Font_Data_Chunk?  Image_Data_Chunk?  Setup_Data_Chunk |
        Font_Data_Chunk?  Setup_Data_Chunk Sequence_Data_Chunk+  Image_Data_Chunk?  |
        Font_Data_Chunk?  Setup_Data_Chunk  Image_Data_Chunk?  Sequence_Data_Chunk+ |
        Font_Data_Chunk?  Sequence_Data_Chunk+  Setup_Data_Chunk  Image_Data_Chunk? |
        Font_Data_Chunk?  Sequence_Data_Chunk+  Image_Data_Chunk?  Setup_Data_Chunk |
        Font_Data_Chunk?  Image_Data_Chunk?  Setup_Data_Chunk  Sequence_Data_Chunk+ |
        Font_Data_Chunk?  Image_Data_Chunk?  Sequence_Data_Chunk+  Setup_Data_Chunk |
        Image_Data_Chunk?  Setup_Data_Chunk  Sequence_Data_Chunk+  Font_Data_Chunk? |
        Image_Data_Chunk?  Setup_Data_Chunk  Font_Data_Chunk?  Sequence_Data_Chunk+ |
        Image_Data_Chunk?  Sequence_Data_Chunk+  Setup_Data_Chunk Font_Data_Chunk?  |
        Image_Data_Chunk?  Sequence_Data_Chunk+  Font_Data_Chunk?  Setup_Data_Chunk |
        Image_Data_Chunk?  Font_Data_Chunk?  Setup_Data_Chunk Sequence_Data_Chunk+  |
        Image_Data_Chunk?  Font_Data_Chunk?  Sequence_Data_Chunk+  Setup_Data_Chunk |)

   Format_Type         = BYTE
   Player_Type         = BYTE
   Text_Encode_Type    = BYTE
   Color_Type          = BYTE
   Timebase            = BYTE
   Option_Size         = BYTE
   Option_Data         = BYTE*     (Option_Size バイト)

   Setup_Data_Chunk  = "Grsu" ChunkSize
                (Display_Prameter_Definition_Chunk   Color_Palette_Definition_Chunk? |
                 Color_Palette_Definition_Chunk?   Display_Prameter_Definition_Chunk)
     Diplay_Parameter_Definition_Chunk = “Gdpd” ChunkSize (prmID Value)+
       prmID     = BYTE
       Value     = BYTE

     Color_Palette_Definition_Chunk =”Gcpd” ChunkSize BYTE+

   Sequence_Data_Chunk = "Grsq" ChunkSize
                  (Duration Event)*   Duration?   (¥x00 ¥x00 ¥x00 ¥x00)+
     Duration = BYTE BYTE?
     Event       = BYTE BYTE*
     Event       = Short_Control_Event | Control_Event | Display_Object_Event
        Short_Control_Event   = EventType
        Control_Event         = EventType EventSize EventData
        DisplayObjectEvent    = EventType EventSize LifeTime Coordinates ObjectSubblock+
           EventType    = BYTE
           EventSize    = BYTE BYTE?
           EventData    = BYTE+    (EventSize バイト)
           Coordinates  = Format?   Position   Position
           ObjectSubblock  = SubblockType SubblockSize SubblockBody
              SubblockType    = BYTE
              SubblockSize    = BYTE BYTE?
              SubblockBody    = BYTE+     (SubblockSize バイト)
```

```
Font_Data_Chunk = “Gftd” ChunkSize   (Font_Chunk* | Unicode_Font_Chunk*)
    Font_Chunk =”Ge” [¥x00-¥xff = FontType][¥x00-¥xff = FontSize] ChunkSize
                  EntryNumber FontData*
        EntryNumber    = BYTE BYTE?
        FontData       = BYTE+
        FontData       = Code Size Bitmap
            Code           = BYTE BYTE
            Size           = BYTE BYTE?
            Bitmap         = BYTE+
            Bitmap         = TYPE XYwidth BitmapData
                TYPE           = BYTE
                XYwidth        = BYTE BYTE BYTE   （12bit x 2 = 3byte 表現）
                BitmapData     = (別途§6.5.2 を参照)
    Font_Chunk =”Gu” [¥x00-¥xff = FontType][¥x00-¥xff = FontSize] ChunkSize
                  EntryNumber FontData*
        EntryNumber    = BYTE BYTE?
        FontData       = BYTE+
        FontData       = Size Code Size Bitmap
            Size           = BYTE BYTE?
            Code           = BYTE+
            Size           = BYTE BYTE?
            Bitmap         = BYTE+
            Bitmap         = TYPE XYwidth BitmapData
                TYPE           = BYTE
                XYwidth        = BYTE BYTE BYTE   （12bit x 2 = 3byte 表現）
                BitmapData     = (別途§6.5.2 を参照)

Image_Data_Chunk =”Gimd” ChunkSize Image_Chunk* Link_Chunk*
    Image_Chunk = ”Gig” [¥x00-¥xff] ChunkSize ImageData*
        ImageData = BYTE*
    Link_Chunk = “Gln* [¥x00-¥xff] ChunkSize LinkData*
        LinkData = BYTE*

;------------------------------------------------------------------------------
```

9.4. 標準音色マップ

Score Track Chunk における再生の互換性を保つため、以下のように標準音色マップを規定する。

9.4.1. Handy phone standard

PC#	Name Bank 0x00	0x80	PC#	Name Bank 0x00	0x80	PC#	Name Bank 0x00	0x80	PC#	Name Bank 0x00	0x80
0	GrandPno		32	AcoBass	Sticks	64	SprnoSax	Conga L	96	Rain	
1	BritePno		33	FngrBass	Bass Drum L	65	AltoSax	Timbale H	97	SoundTrk	
2	E.GrandP		34	PickBass	Open Rim Shot	66	TenorSax	Timbale L	98	Crystal	
3	HnkyTonk		35	Fretless	Bass Drum M	67	Bari.Sax	Agogo H	99	Atmosphr	
4	E.Piano1		36	SlapBas1	Bass Drum H	68	Oboe	Agogo L	100	Bright	
5	E.Piano2		37	SlapBas2	Closed Rim Shot	69	Eng.Horn	Cabasa	101	Goblins	
6	Harpsi		38	SynBass1	Snare M	70	Bassoon	Maracas	102	Echoes	
7	Clavi		39	SynBass2	Hand Clap	71	Clarinet	Samba Whistle H	103	Sci-Fi	
8	Celesta		40	Violin	Snare H	72	Piccolo	Samba Whistle L	104	Sitar	
9	Glocken		41	Viola	Floor Tom L	73	Flute	Guiro Short	105	Banjo	
10	MusicBox		42	Cello	Hi-Hat Closed	74	Recorder	Guiro Long	106	Shamisen	
11	Vibes		43	Contrabs	Floor Tom H	75	PanFlute	Claves	107	Koto	
12	Marimba		44	TremStr	Hi-Hat Pedal	76	Bottle	Wood Block H	108	Kalimba	
13	Xylophon		45	PizzStr	Low Tom	77	Shakhchi	Wood Block L	109	Bagpipe	
14	TubulBel		46	Harp	Hi-Hat Open	78	Whistle	Cuica Mute	110	Fiddle	
15	Dulcimar		47	Timpani	Mid Tom L	79	Ocarina	Cuica Open	111	Shanai	
16	DrawOrgn		48	Strings1	Mid Tom H	80	SquareLd	Triangle Mute	112	TnklBell	
17	PercOrgn		49	Strings2	Crash Cymbal 1	81	SawLead	Triangle Open	113	Agogo	
18	RockOrgn		50	Syn.Str1	High Tom	82	CaliopLd	Shaker	114	SteelDrm	
19	ChrchOrg		51	Syn.Str2	Ride Cymbal 1	83	ChiffLd	Jingle Bell	115	WoodBlk	
20	ReedOrgn		52	ChoirAah	Chinese Cymbal	84	CharanLd	Belltree	116	TaikoDrm	
21	Acordion		53	VoiceOoh	Ride Cymbal Cup	85	VoiceLd		117	MelodTom	
22	Harmnica		54	SynVoice	Tamboulin	86	FifthLd		118	Syn.Drum	
23	TangoAcd		55	Orch.Hit	Splash Cymbal	87	Bass&Ld		119	RevCymbl	
24	NylonGtr	SeqClick H	56	Trumpet	Cowbell	88	NewAgePd		120	FretNoiz	
25	SteelGtr	Brush Tap	57	Trombone	Crash Cymbal 2	89	WarmPad		121	BrthNoiz	
26	JazzGtr	Brush Swirl L	58	Tuba	Vibraslap	90	PolySyPd		122	SeaShore	
27	CleanGtr	Brush Slap	59	Mute.Trp	Ride Cymbal 2	91	ChoirPad		123	Tweet	
28	Mute.G.tr	Brush Swirl H	60	Fr.Horn	Bongo H	92	BowedPad		124	Telphone	
29	Ovrdrive	Snare Roll	61	BrasSect	Bongo L	93	MetalPad		125	Helicptr	
30	Dist.Gtr	Castanet	62	SynBras1	Conga H Mute	94	HaloPad		126	Applause	
31	GtrHarmo	Snare L	63	SynBras2	Conga H Open	95	SweepPad		127	Gunshot	

9.4.2. Mobile standard

normal voice

PC#	Name	PC#	Name	PC#	Name	PC#	Name
0	GrandPno	32	AcoBass	64	SprnoSax	96	Rain
1	BritePno	33	FngrBass	65	AltoSax	97	SoundTrk
2	E.GrandP	34	PickBass	66	TenorSax	98	Crystal
3	HnkyTonk	35	Fretless	67	Bari.Sax	99	Atmosphr
4	E.Piano1	36	SlapBas1	68	Oboe	100	Bright
5	E.Piano2	37	SlapBas2	69	Eng.Horn	101	Goblins
6	Harpsi	38	SynBass1	70	Bassoon	102	Echoes
7	Clavi	39	SynBass2	71	Clarinet	103	Sci-Fi
8	Celesta	40	Violin	72	Piccolo	104	Sitar
9	Glocken	41	Viola	73	Flute	105	Banjo
10	MusicBox	42	Cello	74	Recorder	106	Shamisen
11	Vibes	43	Contrabs	75	PanFlute	107	Koto
12	Marimba	44	TremStr	76	Bottle	108	Kalimba
13	Xylophon	45	PizzStr	77	Shakhchi	109	Bagpipe
14	TubulBel	46	Harp	78	Whistle	110	Fiddle
15	Dulcimar	47	Timpani	79	Ocarina	111	Shanai
16	DrawOrgn	48	Strings1	80	SquareLd	112	TnklBell
17	PercOrgn	49	Strings2	81	SawLead	113	Agogo
18	RockOrgn	50	Syn.Str1	82	CaliopLd	114	SteelDrm
19	ChrchOrg	51	Syn.Str2	83	ChiffLd	115	WoodBlk
20	ReedOrgn	52	ChoirAah	84	CharanLd	116	TaikoDrm
21	Acordion	53	VoiceOoh	85	VoiceLd	117	MelodTom
22	Harmnica	54	SynVoice	86	FifthLd	118	Syn.Drum
23	TangoAcd	55	Orch.Hit	87	Bass&Ld	119	RevCymbl
24	NylonGtr	56	Trumpet	88	NewAgePd	120	FretNoiz
25	SteelGtr	57	Trombone	89	WarmPad	121	BrthNoiz
26	JazzGtr	58	Tuba	90	PolySyPd	122	SeaShore
27	CleanGtr	59	Mute.Trp	91	ChoirPad	123	Tweet
28	Mute.G.tr	60	Fr.Horn	92	BowedPad	124	Telphone
29	Ovrdrive	61	BrasSect	93	MetalPad	125	Helicptr
30	Dist.Gtr	62	SynBras1	94	HaloPad	126	Applause
31	GtrHarmo	63	SynBras2	95	SweepPad	127	Gunshot

drum voice

note#	Name
32	
33	
34	
35	Bass Drum M
36	Bass Drum H
37	Closed Rim Shot
38	Snare M
39	Hand Clap
40	Snare H
41	Floor Tom L
42	Hi-Hat Closed
43	Floor Tom H
44	Hi-Hat Pedal
45	Low Tom
46	Hi-Hat Open
47	Mid Tom L
48	Mid Tom H
49	Crash Cymbal
50	High Tom
51	Ride Cymbal 1
52	Chinese Cymbal
53	RideCymbal Cup
54	Tamboulin
55	Splash Cymbal
56	Cowbell
57	Crash Cymbal 2
58	Vibraslap
59	Ride Cymbal 2
60	Bongo H
61	Bongo L
62	Conga H Mute
63	Conga H Open

note#	Name
64	Conga L
65	Timbale H
66	Timbale L
67	Agogo H
68	Agogo L
69	Cabasa
70	Maracas
71	Samba Whistle H
72	Samba Whistle L
73	Guiro Short
74	Guiro Long
75	Claves
76	Wood Block H
77	Wood Block L
78	Cuica Mute
79	Cuica Open
80	Triangle Mute
81	Triangle Open
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	
92	
93	
94	
95	